Treball de Fi de Màster

Master's degree in Automatic Control and Robotics

Nonlinear Model Predictive Control of a Quadrotor

MEMÒRIA

Autor: Emilio Melgarejo Hernández

Director: Vicenç Puig Cayuela

Convocatòria: Setembre 2016

Escola Tècnica Superior d'Enginyeria Industrial de Barcelona

Abstract

One of the most important features of a quadrotor in order to properly work, generally in some sort of path tracking, is to have a suitable control. This thesis will approach the problem of controlling a quadrotor applying the control technique known as Nonlinear Model Predictive Control.
First, this control should guarantee stability and feasibility, and then the control parameters are tuned to obtain the better possible performance. Proper simulations will be performed by selecting different situations in terms of path tracking references, as well as in terms of the accuracy level of the control model with respect to the real system represented by a high-fidelity model.
Additionally, the requirements for the controller to work in real time will be explored and discussed.

Contents

ETSEIB

List of Figures

ETSEIB

List of Tables

Chapter 1

Introduction

1.1 Motivation

Quadrotors have become, in recent years, a popular and useful tool for a different range of activities. For example, its use for surveillance military, as well as commercial uses to take images of certain landscapes, due to the reduction in costs with respect to a manned vehicle.
Recently, companies (such as Amazon) have taken interest in the possibility of performing delivery operations using quadrotors. It has been also used recently with success for operations of delivery of medical supplies in Germany by a joint project by DHL, Microdrones and RWTH Aachen University, together with other German authorities.
The benefits they provide in terms of maneuverability and simplicity of the mechanics make them an attractive option in order to explore new ideas and possibilities of application. For that reason, the quadrotor is recently having a high focus in terms of research platform, with the aim of providing new uses and/or improvements by developing or testing techniques ranging from computer vision and pattern recognition(such as [8], with the aim of helping against forest fires) to several control techniques. The last one, in fact, has seen large focus in the research, since the control of the quadrotor is the basis of its performance, regardless of the application it is destined to.
For that reason, several approaches have been tested for the control of quadrotors in several articles, such as such as PID and LQR control in [12]. [4], optimized PID in [7], MPC applied to a quadrotor modelled as a Piecewise Affine system in [10], [9], or the use explicit MPC in [9] with a similar piecewise linearization.
Nonlinear control techniques such as Sliding-mode control and Backstepping are applied

in [3]. Also, feedback linearization with a PD controller for the translational subsystem and a backstepping-based nonlinear controller for the rotational subsystem of the quadrotor is applied in [1].

The use of linear control techniques provide powerful and well developed methods to control systems. However, when applied to a nonlinear system, it makes the control restricted to the neighbourhood of the linearization point.

Nonlinear techniques, on the other hand, may make the controller able to work at larger opertaion ranges. And additionally to the advantage that nonlinear control provides in the operating range, nonlinear model predictive control (NMPC), like MPC in general, has the nice property of being able to handle constraints applied to the states and inputs during the computation of a suitable feedback law.

1.2 Objectives

The goal of this work is to explore the use of Nonlinear Model Predictive Control (NMPC) applied to the control of a quadrotor in order to perform path tracking.

The work will be centered on the simulation aspect, carried out with MATLAB, since the use of NMPC requires a high computational effort, therefore making it difficult to implement in a real-time application (which is the reason of its difficult application in the field of quadrotors nowadays).

Therefore, in this thesis it will be used a mathematical model for the quadrotor dynamics which will be based on the work of Bouabdallah in [2]. Using this mathematical model with some simplifications, a control model will be extracted to be used by the NMPC controller in the Optimal Control Problem (OCP).

Afterwards, the controller will be designed. It will be showed the approach of using two control layers taking profit of the linear dynamics of the motors in order to reduce the computational burden.

Then, it will be discussed the benefits and drawbacks of using some different schemes to implement the NMPC, and after choosing the most suitable the NMPC problem will be defined.

Next to this, a path planning Dijkstra algorithm for a 3D map will be implemented in order to provide the controller with suitable reference inputs for the path tracking problem.

Finally, the simulation results will be focused in the following objectives:

1. Using a simple reference input, analyze stability, feasibility and performance for different schemes and configurations, while tuning it in order to reduce computational time and acquire good performance.

2. Determine the maximum operational velocity at which the controller is able to properly control the quadrotor and the configuration parameters that allow to maintain good performance.

3. Computing a complete path for a considered 3D map, analyze the performance of the previous obtained schemes and configurations for this case, as well as the effect of the map complexity in the stability and feasibility of the NMPC.

4. Analyze stability and performance of the controller considering a real model containing unmodelled dynamics and perturbations.

5. Finally, determine the configuration parameters and operation conditions that would allow to obtain a real-time control.

1.3 Outline of the Thesis

The structure of this thesis consists in the following chapters:

1.3.1 Chapter 2: System Model

In this chapter, the dynamics of the system are provided, as well as the corresponding simplifications in order to select the contol-oriented model for the quadrotor rigid-body. Additionally, the dynamic model of the motors is selected, and the procedure to choose the suitable motor parameters is discussed.

1.3.2 Chapter 3: Controller Design

In this chapter, the selection of a two-layer controller structure is stated, as well as the design procedure of the corresponding controllers.

1.3.3 Chapter 4: Results

This chapter presents the results of the performed simulations, in which the procedure of tuning the controller is showed, as well as results regarding the maximum possible linear

velocity for the quadrotor, the controller's performance in front of modelling errors and disturbances, and the necessary configuration in order to obtain real-time execution.

1.3.4 Chapter 5: Environmental Impact

In this chapter, the general effect of the use of quadrotors to the environment is discussed, with its possible advantages and drawbacks.

1.3.5 Chapter 6: Economic Cost

This chapter shows the economic cost resulting of the realization of this project.

1.3.6 Chapter 7: Conclusions and Future work

This chapter will discuss the thoughts on the final state of the project, as well as the possibilities regarding additional features and improvements to add.

Chapter 2

System Model

The quadrotor system can be modelled as a rigid body connecting in a cross shape the four propellers providing the vertical force at each end of it, namely the motors. The configuration of the motors is such that the front and rear motors rotate at the complementary direction of the lateral ones.

The quadrotor makes use of a combination of input velocities to the different rotors in order to control pitch, roll and yaw. It can be visualized in the following images:

Altitude control The altitude of the quadrotor is controlled by providing the same angular speed to the four rotors.

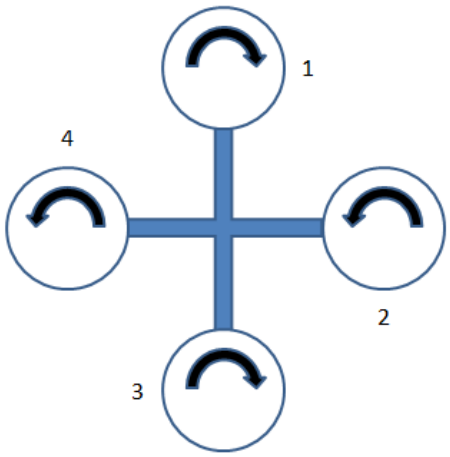

Figure 2.1: Altitude.

Pitch control The pitch angle is controlled by providing a difference in the angular speeds of rotors 2 and 4.

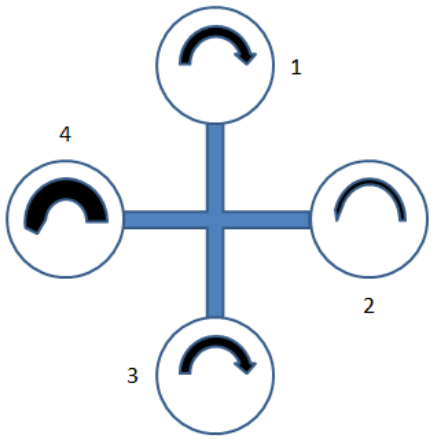

Figure 2.2: Pitch control.

Roll control The pitch angle is controlled by providing a difference in the angular speeds of rotors 1 and 3.

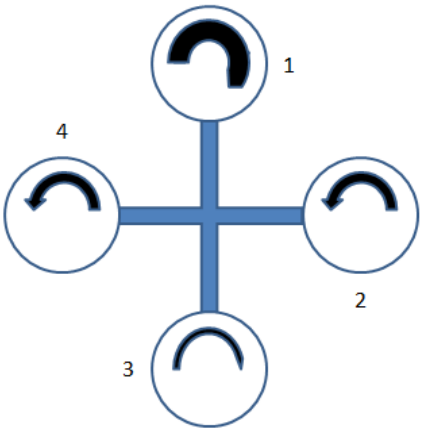

Figure 2.3: Roll control.

Yaw control The pitch angle is controlled by providing a difference in the angular speeds of between the rotors of different direction of rotation.

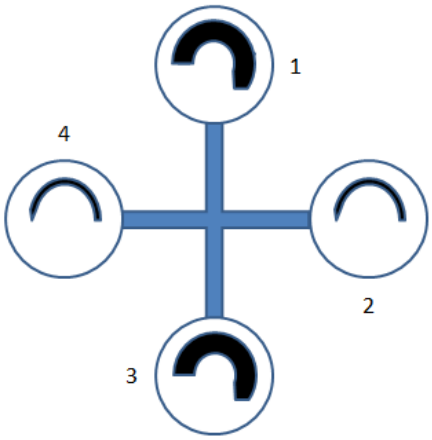

Figure 2.4: Yaw control.

Now, before starting with the description of the system dynamics, we need to define the body fixed frame coordinates. This can be seen in figure 2.5, where ϕ, θ and ψ are the pitch, roll and yaw angles, respectively.

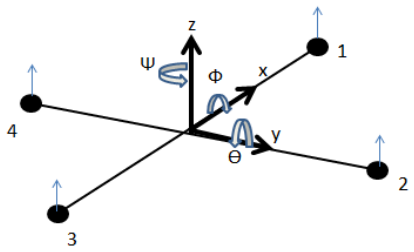

Figure 2.5: Coordinate system of the body fixed frame

2.1 Dynamics of the rigid body

2.1.1 Aerodynamic forces

The aerodynamic forces are derived by Gary Fay in [5] and presented in a summarized way by Bouabdallah in [2].

Thrust force. Thrust forces are those acting on the blades of the rotor parallel to the rotor shaft

$$T = C_T \rho A (\omega R_{rad})^2 \tag{2.1}$$

where C_T is the thrust coefficient, ρ is the air density, A is the propeller disk area, ω is the propeller angular speed and R_{rad} is the propeller disk radius.

Hub force. Hub forces are those acting on the blades of the rotor perpendicular to the rotor shaft (parallel to the plane of the blades)

$$H = C_H \rho A (\omega R_{rad})^2 \tag{2.2}$$

where C_H is the hub coefficient.

Drag moment. Drag moments affect the propeller as a combination of the friction affecting the blades and the angle of attack required to produce lift.

$$Q_m = C_Q \rho A (\omega R_{rad})^2 R_{rad} \tag{2.3}$$

where C_Q is the hub coefficient.

Rolling moment. The rolling moment in the propellers is mainly produced due to a difference in air pressure in the advancing and retreating blade when the quadrotor is on motion.

$$R = C_R \rho A (\omega R_{rad})^2 R_{rad} \tag{2.4}$$

where C_R is the hub coefficient.

2.1.2 General moments and forces

Bouabdallah considers in [2] the following physical effects that describes the motion of the quadrotor by means of the forces and torques produced by said effects:

Rolling moments:

- Body gyroscopic effect:

$$\dot{\theta}\dot{\psi}(I_{yy} - I_{zz}) \tag{2.5}$$

- Propeller gyroscopic effect:

$$J_r\dot{\theta}\Omega_r \tag{2.6}$$

  With Ω_r defined as:

$$\Omega_r = (-1)^{i+1}(\sum_{i=1}^{4}\omega_i) \tag{2.7}$$

- Roll actuators action:

$$l(-T_2 + T_4) \tag{2.8}$$

- Hub moment due to sideward flight:

$$h(\sum_{i=1}^{4}H_{yi}) \tag{2.9}$$

- Rolling moment due to forward flight:

$$(-1)^{i+1}(\sum_{i=1}^{4}R_{xi}) \tag{2.10}$$

Pitching moments:

- Body gyroscopic effect:

$$\dot{\phi}\dot{\psi}(I_{zz} - I_{xx}) \tag{2.11}$$

- Propeller gyroscopic effect:

$$J_r\dot{\phi}\Omega_r \tag{2.12}$$

- Pitch actuators action:

$$l(T_1 - T_3) \tag{2.13}$$

- Hub moment do to forward flight:

$$h(\sum_{i=1}^{4} H_{xi}) \tag{2.14}$$

- Rolling moment due to sideward flight:

$$(-1)^{i+1}(\sum_{i=1}^{4} R_{myi}) \tag{2.15}$$

Yawing moments:

- Body gyroscopic effect:

$$\dot{\theta}\dot{\phi}(I_{xx} - I_{yy}) \tag{2.16}$$

- Inertial counter-torque:

$$J_r\dot{\Omega}_r \tag{2.17}$$

- Counter-torque balance:

$$(-1)^{i}(\sum_{i=1}^{4} Q_i) \tag{2.18}$$

- Hub force unbalance in forward flight:

$$l(H_{x2} - H_{x4}) \tag{2.19}$$

- Hub force unbalance in sideward flight:

$$l(-H_{y1} + H_{y3}) \tag{2.20}$$

Forces along x axis:

- Actuators action:

$$(s\psi s\phi + c\psi s\theta c\phi)(\sum_{i=1}^{4} T_i) \tag{2.21}$$

- Hub force in x axis:

$$-\sum_{i=1}^{4} H_{xi} \tag{2.22}$$

- Friction:

$$\frac{1}{2} C_x A_c \rho \dot{x} |\dot{x}| \tag{2.23}$$

Forces along y axis:

- Actuators action:

$$(-c\psi s\phi + s\psi s\theta c\phi)(\sum_{i=1}^{4} T_i) \tag{2.24}$$

- Hub force in y axis:

$$-\sum_{i=1}^{4} H_{yi} \tag{2.25}$$

- Friction:

$$\frac{1}{2} C_y A_c \rho \dot{y} |\dot{y}| \tag{2.26}$$

Forces along z axis:

- Actuators action:

$$c\psi c\phi(\sum_{i=1}^{4} T_i) \tag{2.27}$$

- Weight:

$$mg \tag{2.28}$$

The parameter l is the horizontal distance between the center of gravity and the propeller, h is the vertical distance from the center of gravity to the propeller, J_r is the propeller inertia, I_{xx} is the quadrotor inertia around x-axis, I_{yy} is the quadrotor inertia around y-axis, I_{zz} is the quadrotor inertia around z-axis, C_x is the friction coefficient affecting the movement in the x-axis, C_y is the friction coefficient for y-axis and A_c is the characteristic contact area between the quadrotor and the wind flow.
Finally, under the assumption that the Body reference frame (B) coincides with the center of mass of the rigid body, the equations of motion are derived in [2] using the Newton-Euler method and the general forces and torques described above.

$$I_{xx}\ddot{\phi} = \dot{\theta}\dot{\psi}(I_{yy} - I_{zz}) + J_r\dot{\theta}\Omega_r + l(-T_2 + T_4) - h(\sum_{i=1}^{4} H_{yi}) + (-1)^{i+1}(\sum_{i=1}^{4} R_{mxi}) \tag{2.29}$$

$$I_{yy}\ddot{\theta} = \dot{\phi}\dot{\psi}(I_{zz} - I_{xx}) - J_r\dot{\phi}\Omega_r + l(T_1 - T_3) + h(\sum_{i=1}^{4} H_{xi}) + (-1)^{i+1}(\sum_{i=1}^{4} R_{myi})$$

$$I_{zz}\ddot{\psi} = \dot{\theta}\dot{\phi}(I_{xx} - I_{yy}) + J_r\dot{\Omega}_r + (-1)^{i}(\sum_{i=1}^{4} Q_i) + l(H_{x2} - H_{x4}) + l(-H_{y1} + H_{y3})$$

$$m\ddot{z} = mg - c\psi c\phi(\sum_{i=1}^{4} T_i)$$

$$m\ddot{y} = (-c\psi s\phi + s\psi s\theta c\phi)(\sum_{i=1}^{4} T_i) - \sum_{i=1}^{4} H_{yi} - \frac{1}{2}C_y A_c \rho \dot{y}|\dot{y}|$$

$$m\ddot{x} = (s\psi s\phi + c\psi s\theta c\phi)(\sum_{i=1}^{4} T_i) - \sum_{i=1}^{4} H_{xi} - \frac{1}{2}C_x A_c \rho \dot{x}|\dot{x}|$$

At this point, some simplifications for the simulation model are considered by the author in [2]. Basically, it is assumed that the hub and rolling moments can be neglected, as well as the friction forces affecting the structure of the quadrotor.

This is usually done with the aim of reducing the computational cost of the controller and in order to be able to meet the required real time constraints of the control. Still, it is not stated by the author the specific reasons for choosing those dynamics to be neglected.

However, it can be appreciated that the rolling moment has a minimal effect, if any, in the moments affecting the quadrotor due to the reason that, on a quadrotor, the rolling moment produced in a propeller during forward or sidewards flight is compensated by the complementary propeller.

This effect exists and can be notable in the case of aircrafts using a single propeller (such as an helicopter) given that, in a forward flight (or sidewards) the advancing and retreating blade (assuming the same angle of incidence) will have different air pressure at the section of the blades, therefore providing different lift forces, which will cause the device to roll towards the retreating blade

The hub moment is produced by the H-force acting parallel to the plane of the rotors. The H-force, as explained in [13] (Chapter 3), consists in an effect produced by two drag forces acting on the retreating and advancing blade of the propeller, the profile drag and the in-

duced drag.

The profile drag is produced as a consequence as the frictional forces acting on the moving blades of the propeller. The induced drag, on the other hand, is the horizontal force acting on the moving blade as a result of the existing angle of attack (which is necessary to produce lift).

As it might seem that over a complete revolution of the propeller these forces are compensated, and the resultant force is zero. Actually, when the aircraft is moving, there exists a difference in air pressure at the advancing and retreating blade (with respect to the movement direction). This, being at the same time responsible for the rolling moment in a propeller, also creates an unbalance between the drag forces acting on the retreating and advancing blade such that there exists a residual backwards force (the H-force). This force is, as stated in [13], very small, even compared to the parasite drag, and thus the reason for it to be neglected.

Finally, the friction forces acting on the non-lifting parts of the aircraft, also called parasite drag, might have been neglected by Bouabdallah because of the fact that, although these forces may have a notable magnitude for high speeds, they have, on the other hand, a small effect at low speeds, which is usually the operating range at which a quadrotor works, and thus the reason for this effect to be neglected. However, it could be advisable to establish bounds to the velocities of the quadrotor when defining the Optimal Control Problem to make this effect have a low impact in the real model.

Additionally, the propeller's gyroscopic effect will be neglected, too.

Therefore, taking into account these simplifications, and performing the corresponding discretization via Euler method with a sampling time T_s, the contol-oriented model becomes the following one:

$$
\begin{aligned}
x_1(k+1) &= x_1(k) + T_s x_2(k) \\
x_2(k+1) &= x_2(k) + T_s a_1 x_4(k) x_6(k) + T_s b_1 U_2(k) \\
x_3(k+1) &= x_3(k) + T_s x_4(k) \\
x_4(k+1) &= x_4(k) + T_s a_3 x_2(k) x_6(k) + T_s b_2 U_3(k) \\
x_5(k+1) &= x_5(k) + T_s x_6(k) \\
x_6(k+1) &= x_6(k) + T_s a_5 x_4(k) x_2(k) + T_s b_3 U_4(k) \\
x_7(k+1) &= x_7(k) + T_s x_8(k) \\
x_8(k+1) &= x_8(k) + T_s g - \frac{T_s}{m} c x_5(k) x_1(k) U_1(k) \\
x_9(k+1) &= x_9(k) + T_s x_{10}(k) \\
x_{10}(k+1) &= x_{10}(k) + \frac{T_s}{m}(-c x_5(k) s x_1(k) + s x_5(k) s x_3(k) c x_1(k)) U_1(k) \\
x_{11}(k+1) &= x_{11}(k) + T_s x_{12}(k) \\
x_{12}(k+1) &= x_{12}(k) + \frac{T_s}{m}(s x_5(k) s x_1(k) + c x_5(k) s x_3(k) c x_1(k)) U_1(k)
\end{aligned}
\tag{2.30}
$$

with

$$
\begin{aligned}
a_1 &= \frac{I_{yy} - I_{zz}}{I_{xx}} \\
a_2 &= \frac{J_r}{I_{xx}} \\
a_3 &= \frac{I_{zz} - I_{xx}}{I_{yy}} \\
a_4 &= \frac{J_r}{I_{yy}} \\
a_5 &= \frac{I_{xx} - I_{yy}}{I_{zz}} \\
b_1 &= \frac{l}{I_{xx}} \\
b_2 &= \frac{l}{I_{yy}} \\
b_3 &= \frac{1}{I_{zz}}
\end{aligned}
\tag{2.31}
$$

Considering the inputs to be:

$$\begin{aligned} U_1 &= b(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \\ U_2 &= b(-\omega_2^2 + \omega_4^2) \\ U_3 &= b(\omega_1^2 - \omega_3^2) \\ U_4 &= d(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2) \end{aligned} \tag{2.32}$$

and the state variables to be:

$$\begin{aligned} X &= \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} & x_{12} \end{bmatrix}^T \\ &= \begin{bmatrix} \phi & \dot{\phi} & \theta & \dot{\theta} & \psi & \dot{\psi} & z & \dot{z} & y & \dot{y} & x & \dot{x} \end{bmatrix}^T \end{aligned} \tag{2.33}$$

2.2 Motor model

The previous set of equations does give the dynamics of the rigid body considering the four torques affecting the quadrotor as inputs, and, by means (2.32) we can obtain the corresponding rotor angular speeds. However, for the quadrotor, the actual control inputs are the voltages applied to the motors that are placed to produce the torques. For that reason, we need to consider a model for the motors. Usually, there are two main options. As one of them, we could consider that the dynamics of the motors are fast enough so the relation between the angular speed and the voltage is simply proportional (so no dynamics are considered). Alternatively, we could provide a dynamic model of the motors to be more accurate, although that would increase the computational cost of the problem.
In this work, we will consider a second order dynamic model for each motor, as follows:

$$\ddot{\omega} = -\frac{R_m J_m + B_m L_m}{L_m J_m}\dot{\omega} + \frac{B_m R_m + K_m^2}{J_m L_m}\omega + \frac{K_m}{J_m L_m} V_{in} \tag{2.34}$$

As for the parameters of the motor, in [2] the author performs parametric identification of the motors chosen, using output data to obtain a first order approximation. However, since the considered model is a closed-loop system with a reference exogenous input, and we need to manipulate the control input to close the loop of the motors, this identified model is not useful for this case.

For that reason, a quick search of a suitable motor to be used in our quadrotor has to be done, in order to provide a realistic model for the simulation and the controller.
To select a suitable motor, some variables from the quadrotor structure, propellers and motors have to be taken into account:

- Total weight of the quadrotor
- Propeller inertia
- Propeller drag moment
- Motor inertia J_m
- Motor inductance L_m
- Motor friction coefficient B_m
- Internal electric resistance R_m

Knowing the total weight of the quadrotor, we can dimension the motors so that their nominal speed provide a thrust two times that of the necessary for hovering. By considering this, we make sure that the motors will provide the quadrotor enough maneuverability to ascend and descend. To have it clear, we cannot dimension the motors by equaling the maximum thrust that can be provided via the motor speeds and the hover thrust, given that, in this way, the motors would have no room to increase thrust in order to lift the quadrotor. Moreover, in case of calculating it too close to the hover thrust, there would still be a high disparity between the maximum acceleration when ascending and when descending. For that reason, it seems reasonable to consider hover thrust as the 50% of the maximum thrust provided by the motors.
From (2.29) we have:

$$\begin{aligned} m\ddot{z} &= mg - c\psi c\phi U_1 \\ m\ddot{z} &= mg - c\psi c\phi b(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \end{aligned} \tag{2.35}$$

Considering hover, then $\psi = 0$, $\phi = 0$, and all four motors will rotate at the same speed ($\omega_i = \omega_h \ \forall i = [1, 4]$) so:

$$4b\omega_h^2 = mg \tag{2.36}$$
$$\omega_h = \sqrt{\frac{mg}{4b}}$$

Therefore, nominal speed (referred as ω_n) will be such that the nominal torque (U_n) is two times that of hover (U_h, with $U_h = 4b\omega_h^2$).

$$U_n = 2U_h \tag{2.37}$$
$$4b\omega_n^2 = 8b\omega_h^2$$
$$\omega_n = = \sqrt{2}\omega_h$$

And by substituting (2.36) in (2.37) we obtain that the nominal motor speed that should be selected is:

$$\omega_n = \sqrt{2\frac{mg}{4b}} \tag{2.38}$$

Note, however, that since ω_n depends on the total mass of the quadrotor, and the introduction of the motors adds an additional mass to the structure, then this computation becomes actually recursive. Still, the worst part of it is that the addition of those masses in the structure would affect to a certain degree the inertia matrix of the structure and the center of mass, whose recomputation would require to do it experimentally. For that reason, the following computations and considerations for the selection of the motors will be made considering that the structure mass m also includes the mass of the motors that will be selected. So, substituting the parameter values from in (2.38), we obtain

$$\omega_n \approx 320 \ \frac{\text{rad}}{\text{s}} \tag{2.39}$$

Now, what will be done is to explore commercial DC motors focusing on just some specifications.

At this point, it is necessary to determine an additional requirement of the chosen motors, which is the output power. When referring to the output power (or mechanical power), the

magnitude that is given is that of the maximum continuous output power, that is, the maximum power that the motor can provide in a steady state situation.
The power output of a motor is given by the following expression:

$$P_{out} = T\omega \tag{2.40}$$

Being the motor torque (T):

$$T = J_m\dot{\omega} + B_m\omega + T_{load} \tag{2.41}$$

with

$$T_{load} = J_l\dot{\omega} + d\omega^2 \tag{2.42}$$

then we have

$$T = (J_m + J_l)\dot{\omega} + B_m\omega + d\omega^2 \tag{2.43}$$

With J_l being the load inertia (in this case, the propeller inertia), and d being the drag coefficient of the propeller.
Thus, the output power expression becomes:

$$P_{out} = (J_m + J_l)\dot{\omega}\omega + B_m\omega^2 + d\omega^3 \tag{2.44}$$

Note that the effect produced by the propeller drag does not change the time constant of the motor, as it just affects its gain. However, here we are considering it in the expression of the torque, unlike in the motor's model (2.34), because it is a big portion of the power demand in steady state, as we will see.
From the expression of the output power, it can be seen that it is not a continuous value. It will actually have a peak value during the transient response of the motor. However, to select a motor with a suitable output power, the value that has to be looked upon is that of the power demand in steady state ($\dot{\omega} = 0$), which means:

$$P_{out} = B_m\omega^2 + d\omega^3 \tag{2.45}$$

As it can be seen, in steady state, the power demand from the motor and the load is given by the sum of friction and drag effect. In this case, the drag effect of the propeller will be the

main factor determining the required power output, given that the power consumption of a motor in steady state due to its friction is actually quite low.

So, even if not explicitly said by the manufacturer, it is assumed that a higher instantaneous power output can be provided, which is actually required by any motor during its transient response, and we will therefore look for a suitable output power as a function of the maximum speed considering only the power consumption due to the propeller drag effect. Note that we only consider the propeller drag effect due to the fact that it allows to look for an orientative value for the output power without it being dependant on the friction of the same motor that is being looked upon, as well as for the fact that it represents the main part of the power consumption.

So, given that the maximum speed at which the motors will rotate is 320 rad/s, then we can obtain the following output power value:

$$\begin{aligned} P_{out} &= d\omega^3 \\ P_{out} &= 7.5 \cdot 10^{-7} \cdot 320^3 \\ P_{out} &= 24.58 \text{ W} \end{aligned}$$

Additionally, it may be also needed the value of the nominal torque, since for a certain output power, the nominal torque of different motors may vary depending on the nominal speed. So, considering:

$$\begin{aligned} T_{nominal} &= d\omega^2 \\ T_{nominal} &= 76.8 \cdot 10^{-3} Nm \end{aligned} \tag{2.46}$$

both output power and torque due to the propeller drag can be reduced by incorporating a gearbox with gear ratio gr, which, on the other hand, would increase the necessary nominal speed for the motor, thus increasing the power consumption and torque due to the motor friction. However, since the motor friction is most likely considerably lower than the propeller drag, with a low gear ratio it could be possible to effectively reduce the output power necessary from the motors.

That leaves us with the following specifications:

- Settling time as low as possible, taking as a reference the value of 0.01s ($\tau = 2.5$ ms)
- Nominal speed of at least $320 \cdot g_r$ rad/s ($3055.8 \cdot g_r$ rpm)
- Output power of at least $24.58/g_r$ W
- Nominal torque of at least $76.8/g_r$ mNm

The motivation of the low settling time is that, since the control inputs computed for the quadrotor upper layer model are considered to be applied with a zero-order hold, then the motors have to provide a response fast enough so that the error due to the inter-sampling dynamics is minimum.
It is important to note that the DC motor data sheets usually provide the mechanical time constant, which is the result of considering the first order approximation for the dynamics of the motor (that is, omitting the dynamics of the electric circuit due to the inductance L). The first order model would be:

$$\frac{\Omega(s)}{V_{in}(s)} = \frac{K_m}{J_m R_m s + B_m R_m + K_m^2} \tag{2.47}$$

In canonical form:

$$\frac{\Omega(s)}{V_{in}(s)} = \frac{\frac{K_m}{B_m R_m + K_m^2}}{\frac{J_m R_m}{B_m R_m + K_m^2} s + 1} \tag{2.48}$$

And the mechanical time constant would be:

$$\tau_{mech} = \frac{J_m R_m}{R_m B_m + K_m^2} \tag{2.49}$$

This data can be taken as a good approximation to the desired time constant (τ) since the mechanical time constant is usually higher enough than the electrical time constant.
With this expression we can appreciate that the time constant is directly proportional to the inertia of the motor. However, when choosing a motor, it has to be taken into account that the total inertia will not be only that of the motor, but the sum of the motor inertia and the propeller inertia. This means that the equivalent time constant could be higher than that of the motor without load.

In fact, the introduction of a load makes the mechanical constant to effectively become:

$$\tau_{mech} = \frac{J_m R_m}{R_m B_m + K_m^2} + \frac{J_L R_m}{R_m B_m + K_m^2} \tag{2.50}$$

Given that commercial motors are already constructed to have a certain mechanical time constant without load, then we can consider

$$\tau_{mech} = \tau_{mech_m} + \frac{J_L R_m}{R_m B_m + K_m^2} \tag{2.51}$$

where τ_{mech_m} stands for the mechanical time constant of the motor without load.

This means that, when looking for a motor with a certain mechanical time constant, we can minimize the effect of the load in the resultant time constant by choosing a motor that has a K as high as possible. However, the combination of a low time constant for the motor with a high torque constant usually comes at the expense of a higher power demand. For that reason, we will select a motor meeting the specifications stated above and try to compensate any variation in the time constant by applying optimal control to the motors, which may allow us to still obtain a fast enough response. In case of obtaining a low performance with this approach, we will select a more powerful motor.

If a gearbox of gear ratio g_r were included, the load inertia seen by the motor would be divided by g_r, which would also help reducing the impact of the load inertia into the time constant.

Therefore, after analyzing some commercial DC motors, we selected the Maxon DCX 22L @ 12 V which has the following specifications:

$$\omega_n = 10700 \text{ rpm} = 1120.5 \text{ rad/s} \tag{2.52}$$
$$T_{nominal} = 30.5 \text{ mNm} \tau_m = 3.21 \text{ ms}$$
$$P_{out} = 34 \text{ W}$$
$$R_m = 0.335 \ \Omega$$
$$L_m = 0.035 \text{ mH}$$
$$J_m = 9.06 \cdot 10^{-7} \ \frac{\text{Nm}}{\text{rad/s}^2}$$
$$B_m = 2 \cdot 10^{-6} \frac{\text{Nm}}{\text{rad/s}}$$
$$K_m = 9.73 \cdot 10^{-3} \ \frac{\text{Nm}}{\text{A}}$$

This motor, with a gear ratio $g_r = 3$, is able to meet the specifications stated.

Note that it has been selected with a quite higher power than the required by the drag effect of the propeller, which would be, from 2.2, $P_{out_d} = \frac{24.58}{3} = 8.19$ W. However, since the required motor nominal speed is multiplied by the g_r ($\omega_n = 320 \cdot 3 = 960$ rpm/s), then the output power required to cover the motor friction effect goes up. This translates, with the friction coefficient this motor has, into the following output power required:

$$P_{out} = B(\omega g_r)^2 + d\omega^3 P_{out} = B(320 \cdot 3)^2 + d320^3 = 26.42 \text{ W} \tag{2.53}$$

which is covered by the 34 W of output power that the motor is able to provide.

In Figure 2.6 and 2.7, the open-loop output speed of this motor with the considered gear ratio can be seen with and without load, respectively. It is appreciated that the specifications do meet with the dynamic model without load, and how the introduction of the load increases notably the time constant.

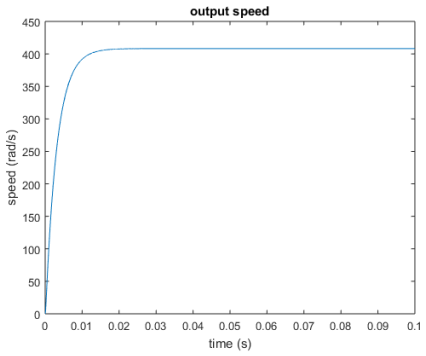

Figure 2.6: Open loop response of the DC motor without the propeller load

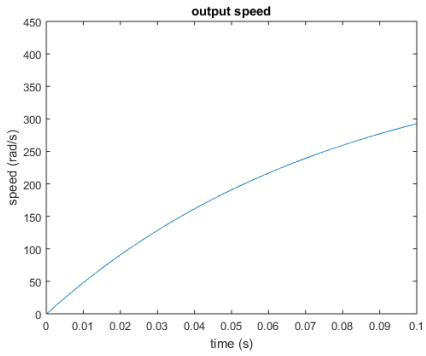

Figure 2.7: Open loop response of the DC motor with the propeller load

In order to compensate for the increase in the time constant caused by the propeller's inertia, optimal control can be applied. In this case, in order to reduce the burden on the already costly computation of the NLP, the optimal control used on the motors is not an MPC, as we will not consider a receding horizon in which we only take the first input value ($u^*(0)$) of the optimal control sequence ($u^*(\cdot)$) to be applied to the real model. In this case, it is more like an optimal regulator, in the sense that, over an optimization horizon, the entire optimal control sequence ($u^*(\cdot)$) is applied without having feedback of the real states.

Chapter 3

Controller design

A general closed-loop system with no disturbances or noise can be represented in a block diagram such as in figure 3.1.

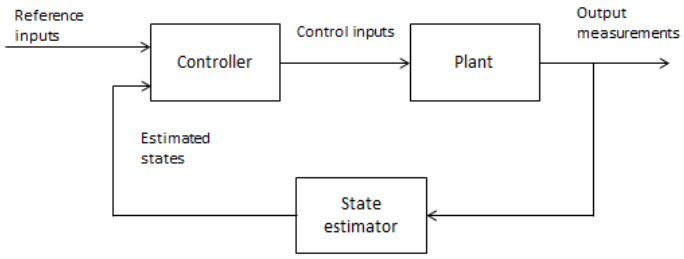

Figure 3.1: General closed-loop system diagram.

In our case, being the controller based in the NMPC method, the full state set of the quadrotor is required, which would make necessary the use of an state estimator. However, it will be assumed that all states of the quadrotor can be accessed and therefore no observer will be designed.

The controller of the quadrotor is designed by using Nonlinear Model Predictive Control. Therefore, the structure of the controller can be depicted in Figure 3.2.

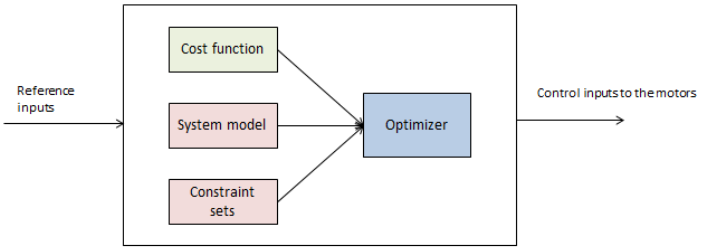

Figure 3.2: NMPC controller diagram.

Including the motor dynamics, being described by the discretization of (2.34), and assuming the four of them have exactly the same dynamics, we have

$$
\begin{aligned}
x_{m1}(k+1) &= x_{m1}(k) + T_{sm}x_{m5}(,k) \\
x_{m5}(k+1) &= T_{sm}\Big(-\Big(\frac{R_m}{L_m}+\frac{B_m}{J_t}\Big)x_{m5}(k) - \frac{B_mR_m+K_m^2}{J_tL_m}x_{m1}(k)\Big) + x_{m5}(k) + T_{sm}\frac{K_m}{J_tL_mg_r}u_{m1}(k) \\
x_{m2}(k+1) &= x_{m2}(k) + T_{sm}x_{m6}(,k) \\
x_{m6}(k+1) &= T_{sm}\Big(-\Big(\frac{R_m}{L_m}+\frac{B_m}{J_t}\Big)x_{m6}(k) - \frac{B_mR_m+K_m^2}{J_tL_m}x_{m2}(k)\Big) + x_{m6}(k) + T_{sm}\frac{K_m}{J_tL_mg_r}u_{m2}(k) \\
x_{m3}(k+1) &= x_{m3}(k) + T_{sm}x_{m7}(,k) \\
x_{m7}(k+1) &= T_{sm}\Big(-\Big(\frac{R_m}{L_m}+\frac{B_m}{J_t}\Big)x_{m7}(k) - \frac{B_mR_m+K_m^2}{J_tL_m}x_{m3}(k)\Big) + x_{m7}(k) + T_{sm}\frac{K_m}{J_tL_mg_r}u_{m3}(k) \\
x_{m4}(k+1) &= x_{m4}(k) + T_{sm}x_{m8}(,k) \\
x_{m5}(k+1) &= T_{sm}\Big(-\Big(\frac{R_m}{L_m}+\frac{B_m}{J_t}\Big)x_{m8}(k) - \frac{B_mR_m+K_m^2}{J_tL_m}x_{m4}(k)\Big) + x_{m8}(k) + T_{sm}\frac{K_m}{J_tL_mg_r}u_{m4}(k)
\end{aligned}
\tag{3.1}
$$

Therefore, introducing it into the NMPC description would lead to a complete description of the model with the following states:

$$
X = \begin{bmatrix} \phi & \dot{\phi} & \theta & \dot{\theta} & \psi & \dot{\psi} & z & \dot{z} & y & \dot{y} & x & \dot{x} & \omega_1 & \omega_2 & \omega_3 & \omega_4 & \dot{\omega}_1 & \dot{\omega}_2 & \dot{\omega}_3 & \dot{\omega}_4 \end{bmatrix}^T \tag{3.2}
$$

and inputs:

$$U = \begin{bmatrix} u_{m1} & u_{m2} & u_{m3} & u_{m4} \end{bmatrix}^T \tag{3.3}$$

However, considering the linear dynamics of the motors, we could optimize the controller structure by separating the nonlinear of the quadrotor rigid body and the linear dynamics of the motors in two different layers, namely, upper controller layer and lower controler layer, respectively. With this approach, we would be able to obtain an NMPC controller with 12 variables instead of 20, and an additional linear controller for the motors, with the evident reduction in computational effort due to the use of specialised controllers for each case.

This can be seen schematically in Figure 3.3.

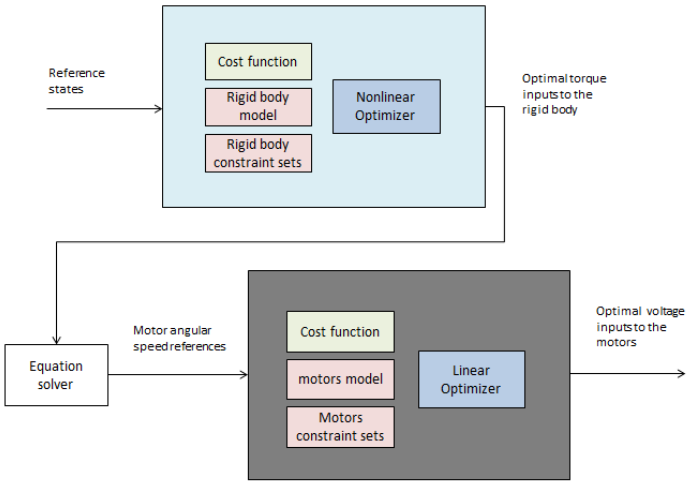

Figure 3.3: Upper and lower layer controllers diagram

The drawback of this approach would be that, given that the motor dynamics are not considered in the NMPC controller, then the optimal input computed by it are the torques that would need to be applied to the rigid body structure of the quadrotor. The problem of this is that, since the NMPC computes the optimal control inputs by considering them to have zero order dynamics (that is, to be steps) allong the sampling time, then we need that the motor controller is able to provide a response fast enough so that the error due to the intersampling dynamics remains low.

3.1 Motor control

Due to the mentioned problem regarding inter-sampling dynamics, it is going to be considered an optimal controller for the motors, as it was already briefly explained at the end of Section 2.2. Basically, choosing an optimal regulator in the spirit of a linear MPC but without considering a receding horizon, it allows us to define the constraint sets of the motors and to obtain an optimal response accordingly that can be, provided the reference step is low enough, fast enough so that the error with respect to the referece remains low even with the consideration of the propeller's inertia into the dynamics of the motor.
Therefore, the controller would consist in solving the Optimal Control Problem for a fixed reference x_m^{ref}.

$$\begin{aligned}
&\text{minimize} \quad J(x_{m0}, u_m(\cdot)) = \sum_{k=0}^{N_{opt}-1} \ell(k, x_{um}(k, x_{m0}), u_m(k)) \\
&\text{with respect to} \quad u_m(\cdot) \\
&\text{subject to} \quad x_{um}(0, x_{m0}) = x_{m0} \\
&\qquad\qquad\quad x_{um}(k+1, x_{m0}) = g(x_{um}(k, x_{m0}), u_m(k)) \\
&\qquad\qquad\quad x_{um}(\cdot, x_{m0}) \in \mathbb{X}_m^N \\
&\qquad\qquad\quad u_m(\cdot) \in \mathbb{U}_m^N
\end{aligned} \tag{3.4}$$

with $J(x_{m0}, u_m(\cdot))$ being the finite horizon cost function, $\ell(k, x_{um}(k, x_{m0}), u_m(k))$ being the running cost function, $u_m(\cdot)$ the input control sequence and $x_{um}(\cdot, x_{m0})$ the state trajectory resultant from said control sequence, with x_{m0} being the initial state.
Note that the optimization horizon N_{opt} is selected such that, assuming that the sampling time of this optimal controller is lower than that of the NMPC controller due to the faster dynamics of the motors, covers a complete sampling time of the upper layer controller.
Then, considering $u_m^*(\cdot)$ to be the optimal control sequence obtained from the solution of the OCP, it would be applied entirely to the motors with the corresponding sampling time.
The choice for the running cost function needs for it to be positive semidefinite. For that reason, we select a quadratic running cost function such as:

$$J(x_m, u_m) = \sum_{k=0}^{N_{opt}-1} (x_m - x_m^{ref})^T P_m (x_m - x_m^{ref}) + u_m^T Q_m u_m \tag{3.5}$$

with P_m being the weighting matrix for the motor states and Q_m the weighting matrix for the control inputs.

Finally, the constraints sets have to be defined. That is done by considering the operational and physical limitations that bound them.

In this case, the only limitation considered is in the angular speed, as it cannot exceed 320 rad/s, according to the output power computations for the motor. Also, the input values are bounded by the phisical limitations related to the input voltage.

Therefore, we define $\mathbb{X}_m$ as

$$\begin{aligned} \omega_1 &\in [0 \;\; 320] \\ \omega_2 &\in [0 \;\; 320] \\ \omega_3 &\in [0 \;\; 320] \\ \omega_4 &\in [0 \;\; 320] \\ \dot{\omega}_1 &\in [-\infty \;\; \infty] \\ \dot{\omega}_2 &\in [-\infty \;\; \infty] \\ \dot{\omega}_3 &\in [-\infty \;\; \infty] \\ \dot{\omega}_4 &\in [-\infty \;\; \infty] \end{aligned} \tag{3.6}$$

and the input constraint set $\mathbb{U}_m$ as

$$\begin{aligned} u_{m1} &\in [0 \;\; 12] \\ u_{m2} &\in [0 \;\; 12] \\ u_{m3} &\in [0 \;\; 12] \\ u_{m4} &\in [0 \;\; 12] \end{aligned} \tag{3.7}$$

In Figure 3.4, it can be seen the optimal response of the motor for a speed reference of 60 rad/s (upper figure) as well as the optimal control inputs (lower figure).

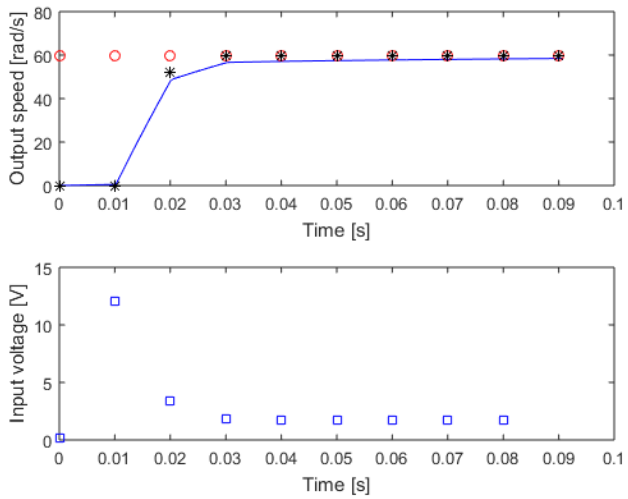

Figure 3.4: Up: optimal state trajectory for the motors with a reference speed of 60 rad/s (Red line: reference. Black asterisks: discrete state trajectory. Blue line: continuous state trajectory). Down: optimal control sequence

As it can be seen, the use of this optimal regulator solves the problem of the increasing time constant when including the propeller inertia.
However, its performance depends on the variation in the reference value, as an increasingly large reference step would make the rate of convergence of optimal response to approach that of the open loop response, thus providing the optimization almost no benefit, or none at all. By doing some simulations, it was decided that the limit in the variation of the reference speed for the motors would be of 60 rad/s (such as that of Figure 3.4. This will be reflected in the bounds for the control inputs of the upper layer of the controller.

3.2 NMPC Definition

Once selected the mathematical model for the quadrotor rigid body, having defined the structure of the controller and designed the motor optimal regulator, the NMPC problem has to be defined.
The structure of a general MPC problem consists in the computation of control inputs that minimize a cost function, subject to a certain set of constraints that includes the control oriented model of the system, over a certain prediction horizon N.
This translates into the following algorithm for a basic NMPC:

1. Measure the state $x(n)$ of the system (or real model, in our case).

2. Set $x_0 = x(n)$, solve the Optimal Control Problem

$$\begin{aligned} &\text{minimize} \quad J(n, x_0, u(\cdot)) = \sum_{k=0}^{N-1} \ell(n+k, x_u(k, x_0), u(k)) \\ &\text{with respect to} \quad u(\cdot) \\ &\text{subject to} \quad x_u(0, x_0) = x_0 \\ &\qquad\qquad x_u(k+1, x_0) = f(x_u(k, x_0), u(k)) \\ &\qquad\qquad x_u(\cdot, x_0) \in \mathbb{X}^N \subseteq \mathbb{R}^{i \times N} \\ &\qquad\qquad u(\cdot) \in \mathbb{U}^N(x_0) \subseteq \mathbb{R}^{j \times N} \end{aligned} \tag{3.8}$$

   where x is the set of states, u the set of inputs, $\mathbb{X}^N$ and $\mathbb{U}^N$ are their respective constraint sets along prediciton horizon N, J_N is the finite horizon cost, ℓ is the running cost function and $x_u(\cdot, x_0)$ is the trajectory corresponding to control sequence $u(\cdot)$ from initial state x_0.

3. Denote the computed optimal control sequence as $u^*(\cdot)$.

4. Select the first term of the optimal control sequence as the feedback value ($\mu_N(n, x(n)) = u^*(0)$)) and apply it to the system in the next sampling period.

Note that the control input constraint set is defined as depending on the initial state ($\mathbb{U}^N(x_0)$, or, in general, $\mathbb{U}(x_0)$). This definition implies that this set may be defined such that $\mathbb{X}$ remains viable for any x_0. However, in our case it will not be done in such a way, holding therefore the equivalency $\mathbb{U}(x_0) \equiv \mathbb{U}$.
Having defined the basic NMPC algorithm, some variations can be considered in the OCP from (3.8), with the aim of increasing performance or enforce stability. Some approaches are proposed in [11] by Grüne and Pannek, such as the use of terminal constraints or the Lyapunov function terminal cost, whose properties regarding stability, as well as some others, are analyzed in detail there.
The use of a terminal constraint set to force stability implies the definition of a set of constraints for the states at the last step of the optimization ($\mathbb{X}_0(n)$ for time varying reference case), forcing it to be equal to the reference value.
So, the equality of the terminal constraint set to the reference:

$$\mathbb{X}_0(n) = x^{ref}(n) \tag{3.9}$$

translates into the following constraint for the OCP:

$$x_u(N, x_0) = x^{ref}(n+N) \tag{3.10}$$

Therefore, considering that this problem will have a time varying reference, then the NMPC with terminal constraints is defined as follows:

$$\begin{aligned} &\text{minimize} \quad J(n, x_0, u(\cdot)) = \sum_{k=0}^{N-1} \ell(n+k, x_u(k, x_0), u(k)) \\ &\text{with respect to} \quad u(\cdot) \in \mathbb{U}_{\mathbb{X}_0}^N(x_0) \\ &\text{subject to} \quad x_u(0, x_0) = x_0, \quad x_u(k+1, x_0) = f(x_u(k, x_0), u(k)) \\ &\qquad\qquad x_u(N, x_0) = x^{ref}(n+N) \end{aligned} \tag{3.11}$$

The approach of using Lyapunov terminal cost function solves the drawback that affects the use of a terminal constraint set in the sense of the previous approach, which is the fact that the system needs to be exactly controlable to $x^{ref}(n)$ in finite time, as well as the fact that in nonlinear and nonconvex problems. The strict endpoint constraint may arise numerical problems in the optimization.
In this case, the OCP is defined as follows:

$$\begin{aligned} &\text{minimize} \quad J(n, x_0, u(\cdot)) = \sum_{k=0}^{N-1} \ell(n+k, x_u(k, x_0), u(k)) + F(n+N, x_u(N, x_0)) \\ &\text{with respect to} \quad u(\cdot) \in \mathbb{U}_{\mathbb{X}_0}^N(x_0) \quad \text{with} \quad x^{ref}(n) \in \mathbb{X}_0(n) \\ &\text{subject to} \quad x_u(0, x_0) = x_0, \quad x_u(k+1, x_0) = f(x_u(k, x_0), u(k)) \\ &\qquad\qquad x_u(N, x_0) = x^{ref}(n+N) \end{aligned} \tag{3.12}$$

where F is the Lyapunov terminal cost function, X_0 is the feasible set, and they are defined

such that they meet the following assumptions:

(1) X_0 is a viable set, that is, for each $x \in \mathbb{X}$ there exists an admissible control value $u_x \in \mathbb{U}(x)$ such that

$$\mathbb{X}_N(n) = \{x_0 \in \mathbb{X} \mid \text{there exists } u(\cdot) \in \mathbb{U}^N(x_0) \text{ with } x_u(N, x_0) \in \mathbb{X}_0(n+N)\} \tag{3.13}$$

holds

(2) The terminal cost function $F : \mathbb{X}_0 \to \mathbb{R}_0^+$ is such that for each $x \in \mathbb{X}$ there exists an admissible control value $u_x \in \mathbb{U}(x)$ so that assumption (**1**) and

$$F(f(x, u_x)) + \ell(x, u_x) \leq F(x) \tag{3.14}$$

hold.

This essentially means that once the states enter the terminal constraint set, the system is guaranteed to be stable. This allows to have an OCP with a relaxed terminal constraint with respect to the endpoint terminal constraint scheme.

However, the computation of both F and X_0, as detailed in [11](Section 5.2) is quite complex, especially in the time varying case, which is the major drawback of this scheme.

Finally, it is also possible that these schemes used in order to enforce stability might be unnecessary, thus obtaining the desired results with an scheme without terminal cost and constraints, with the obvious advantages of it in terms of computation.

Still, other properties might be important to consider, as well as stability, in order to select a scheme to be used in the NMPC. Therefore, Grüne and Pannek [11] perform a comparison between the three schemes:

a NMPC without using terminal cost or constraints.

b NMPC with endpoint terminal contraints.

c NMPC with Lyapunov terminal cost function.

Which are compared in the following terms:

i design

ii stability

iii performance

iv feasibility

v numerical effort

The conclusions extracted could be summarized as follows:
(i) In the design aspect, both schemes (a) and (b) are preferable over (c), since the only thing that has to be designed is the desired reference and the cost function ℓ. On the other hand, on scheme (c) it is necessary to construct the terminal cost function, which is a rather complex task.
Specifically in this problem, being a problem with a time varying reference, scheme (c) is by far outshined by the other two in terms of design, since the construction of F and $\mathbb{X}_0$ is specially complex for this case, as stated in [11](Remark 5.16).
(ii) Regarding stability, there are two main differences between (a) and (b)(c). In schemes (b) and (c), the region of attraction of the reference for the closed-loop solution is limited by the feasible set ($\mathbb{X}_N(n)$ for the time-varying case), being this set defined as:

$$\mathbb{X}_N(n) = \{x_0 \in \mathbb{X} \mid \text{there exists } u(\cdot) \in \mathbb{U}^N(x_0) \text{ with } x_u(N, x_0) \in \mathbb{X}_0(n+N)\} \tag{3.15}$$

which can be read as the set of initial values for which the computed inputs ($u(\cdot)$) yield a state trajectory $x_u(\cdot, x_0)$ whose value at the end of the optimization horizon ($x_u(N, x_0)$) remains inside the time varying terminal constraint set ($\mathbb{X}_0(n+N)$).
On the other hand, scheme (a) may provide, for a fixed N, a larger, or even unbounded region of attraction.
However, for a decreasing N, in schemes (b) and (c) the effect is that the region of attraction of the reference shrinks, while in scheme (a) asymptotic stability may be lost. Therefore, which of the two properties is more advantageous has to be determined case by case.
In this case, specifically, since we can set the initial state of the simulation arbitrarily close to the terminal state at time $n = N$, it might be more advantageous to implement scheme (b) or (c) over (a), regarding this point about stability. However, it could also mean that the variation of the reference (thus, the speed of the quadrotor) would have to be lower than a certain value for a certain prediction horizon, in order to have the initial value at all times inside the feasible set. So, at the end, it would be needed to determine by simulation which scheme provides better results at this point.

(iii) As for performance, schemes (b) and (c) cause, due to the strict terminal conditions, the system states to approach more rapidly to the reference at the expense of a greater control effort, while the unconstrained scheme (a) does not need approach the reference value in such a strict way, which therefore translates in a lower control effort. This means that, as stated by Grüne and Pannek [11], in general it appears that a stronger penalization in the control inputs make scheme (c) to yield better performance, and viceversa.

For this case, since the main objective will be to steer the system to the reference values, even if there is some penalization of the input, schemes (b) and (c) might be better candidates regarding performance. However, since the term "strong penalization" remains quite subjective, at the end it would need to be determined by simulation which of the schemes yield better performance for a certain weighting of the states and inputs.

(iv) Regarding feasibility, it is shown in [11] (Lemma 5.2) that, for schemes (b) and (c), the sets $\mathbb{X}_N$ are recursively feasible, which means that if a given state x is feasible, then its successor state $f(x, \mu_N(x))$ is also feasible, with $\mu_N(x)$ being the feedback law (in general, $\mu_N(x)$ is the first term of the optimal control sequence $u^*(0)$).

$$x \in \mathbb{X}_N(n) \Rightarrow f(x, \mu_N(n, x)) \in \mathbb{X}_N(n) \tag{3.16}$$

The previous statement is true as long as the terminal constraint set $\mathbb{X}_0$ is viable:

$$\forall x \in \mathbb{X}_0(n) \;\; \exists \;\; u \in \mathbb{U} \;\mid\; f(x,u)) \in \mathbb{X}_0(n+1) \tag{3.17}$$

On the other hand, in scheme (a) recursive feasibility can also be obtained on its feasible set, but, unlike in (b) and (c), additional assumptions are needed.

In the case we could compute the feasible sets for both constrained and unconstrained schemes, then both schemes (b) and (c) have the obvious advantage of being able to certainly be feasible by meeting only the assumption about viability of $\mathbb{X}_0$ (which might require the reference to be slowly varying, such as in the case of stability in (ii)). However, given that the means to compute these feasible sets were not found, then the comparison, again, is left to be determined experimentally by simulation.

(v) Finally, with respect to the numerical effort, it is obvious that for a fixed N the constraints in schemes (b) and (c) make them more demanding than (a), particularly the endpoint condition in (b). So, a priori, the preferable scheme to reduce numerical effort would be (a). However, if the use of schemes (b) and (c) helped reduce the prediction horizon N this could

outweight by far the benefits that (a) has for not including the additional constraints. Thus, again, the behaviour of these schemes with respect to the numerical effort would need to be assessed case by case.
This comparison shows that there are two properties for which we can determine the most advantageous scheme for our case, which would be in the design and (probably) performance aspects. As for the rest of them, the respective advantages of the different schemes show that the benefits for our case would need to be determined by simulations.
Still, the logical course of action would be to choose the most simple scheme to design, which would be (a), applying simple modifications, like adding terminal weights.
Also, since scheme (b) is almost as easy as (a) to implement, both of them will be compared during the simulations and its respective advantages in each property discussed above will be determined.
On the other hand, even if scheme (c) might be able to provide benefits, the design aspect outweights them, a priori, by far, thus this scheme won't be further explored unless absolutely necessary (in case both schemes (a) and (b) fail to provide acceptable results).

3.2.1 Defining cost function and constraint sets

Now that we have chosen the schemes for the NMPC problem, it is necessary to define the cost function ℓ, as well as the state constarint set $\mathbb{X}$ and the input constraint set $\mathbb{U}$.
For the design of the cost function, the main desire is to penalize the distance of the states x to the reference $x^{ref}(n)$. Additionally, it can be desirable to penalize as well the use of the control input u and its increment Δu, as it can make the optimization problem to be easier to solve, as well as for the minimization of the energy consumption and stability purposes.
Therefore, we require that for the cost function $\ell(x, u, \Delta u)$ (considering it to be time-invariant)

$$\begin{aligned} &\ell(x^{ref}(n), 0, 0) = \alpha, \\ &\ell(n, x, u) > \alpha \quad \forall x \neq x^{ref}(n), \;\; u \neq 0, \;\; \Delta u \neq 0 \end{aligned} \tag{3.18}$$

holds, where α is some, a priori, unknown lower bound for the cost function. It is not 0 due to the fact that the conditions for it would be that, for $x = xref(n)$ $u = 0$ and $\Delta u = 0$, then $f(x, u) = x^{ref}(n+1)$ which will not be the case due to the dynamics of the system.
This requirement means that ℓ has to be positive semidefinite. Therefore, we may choose a

quadratic cost function of the form:

$$\ell(x,u) = \|x - x^{ref}(n)\|^2 + \|u\|^2 + \|\Delta u\|^2 \tag{3.19}$$

which, applying the corresponding weights to the states and inputs, can be rewritten in matricial form as follows:

$$\ell(x,u) = (x - x^{ref})^T P(x - x^{ref}) + u^T Q u + \Delta u^T R \Delta u \tag{3.20}$$

where $P \subset \mathbb{R}^{nxn}$, $Q \subset \mathbb{R}^{mxm}$ and $R \subset \mathbb{R}^{mxm}$ are the positive semidefinite matrices containing the weights.

Now, having the cost function defined, we need to construct the constraint sets $\mathbb{X}$ and $\mathbb{U}$ to include in the Optimal Control Problem.

Therefore, we will start by defining the state constraint set $\mathbb{X}$. This is defined by the physical and operational limitations of the quadrotor.

$$\begin{aligned}
\phi &\in [-\frac{\pi}{2} \ \ \frac{\pi}{2}] \\
\theta &\in [-\frac{\pi}{2} \ \ \frac{\pi}{2}] \\
\psi &\in [-\frac{\pi}{2} \ \ \frac{\pi}{2}] \\
\dot{\phi} &\in [-\infty \ \ \infty] \\
\dot{\theta} &\in [-\infty \ \ \infty] \\
\dot{\psi} &\in [-\infty \ \ \infty] \\
x &\in [-\infty \ \ \infty] \\
y &\in [-\infty \ \ \infty] \\
z &\in [-\infty \ \ \infty] \\
\dot{x} &\in [-\infty \ \ \infty] \\
\dot{y} &\in [-\infty \ \ \infty] \\
\dot{z} &\in [-\infty \ \ \infty]
\end{aligned} \tag{3.21}$$

Initially, this would be the constraint set $\mathbb{X}$, which would be defined by operational limitations on the pitch and roll (limited by the fact that we do not desire the quadrotor to be

upside-down). This set might be updated on the following sections, in case it is seen as necessary to establish additional bounds to the states.
As for the control input constraint set $\mathbb{U}$, it will be defined such that it is consistent with the maximum and minumum forces and torques provided by the motors, knowing their maximum and minimum angular speed.
That would be:

$$\begin{aligned}
\max\ u_1 &= 4b(\overline{\omega}^2) \\
\max\ u_2 &= b(\overline{\omega}^2 - \underline{\omega}^2) \\
\max\ u_3 &= b(\overline{\omega}^2 - \underline{\omega}^2) \\
\max\ u_4 &= 2d(\overline{\omega}^2 - \underline{\omega}^2) \\
\min\ u_1 &= 4b(\underline{\omega}^2) \\
\min\ u_2 &= b(\underline{\omega}^2 - \overline{\omega}^2) \\
\min\ u_3 &= b(\underline{\omega}^2 - \overline{\omega}^2) \\
\min\ u_4 &= 2d(\underline{\omega}^2 - \overline{\omega}^2)
\end{aligned} \tag{3.22}$$

Therefore, knowing that $\omega_i \in [\underline{\omega}\ \ \overline{\omega}] = [0\ \ 320]$ then we can compute the bounds for the control inputs u_i.

$$\begin{aligned}
u_1 &\in [0\ \ 12.8205] \\
u_2 &\in [-3.2051\ \ 3.2051] \\
u_3 &\in [-3.2051\ \ 3.2051] \\
u_4 &\in [-0.1536\ \ 0.1536]
\end{aligned} \tag{3.23}$$

Note that, even if the constraint set $\mathbb{U}$ is computed by using the maximum values of ω_i, that does not imply that a set of inputs $u \in \mathbb{U}$ such that $x_m \notin \mathbb{X}_m$ does not exist, with x_m being the motor states and $\mathbb{X}_m$ the motor state constraint set.
Regarding the constraint set for the variation in the control inputs $\Delta u \in \mathbb{D}$, it can be defined in the same manner as $\mathbb{U}$, which is taking into account the maximum variation allowed for the motor angular speeds ($\Delta\omega$). Since u_i are not linear functions with respect to ω_i, then we need to define first $\Delta u_i(k) = u_i(k+1) - u_i(k)$ as:

$$\begin{aligned}
\Delta u_1(k) &= b(\omega_1^2(k+1) - \omega_1^2(k) + \omega_2^2(k+1) - \omega_2^2(k)\omega_3^2(k+1) - \omega_3^2(k) + \omega_4^2(k+1) - \omega_4^2(k)) \\
\Delta u_2(k) &= b(\omega_4^2(k+1) - \omega_4^2(k) - (\omega_2^2(k+1) - \omega_4^2(k))) \\
\Delta u_3(k) &= b(\omega_1^2(k+1) - \omega_1^2(k) - (\omega_3^2(k+1) - \omega_3^2(k))) \\
\Delta u_4(k) &= d(-(\omega_1^2(k+1) - \omega_1^2(k)) + (\omega_2^2(k+1) - \omega_2^2(k)) - (\omega_3^2(k+1) - \omega_3^2(k)) + (\omega_4^2(k+1) - \omega_4^2(k)))
\end{aligned} \tag{3.24}$$

Now, we will consider $\omega_i(k)$ to be:

$$\omega_i(k) = \omega_i(k+1) - \Delta\omega_i(k) \tag{3.25}$$

Then, substituting (3.25) in (3.24), we obtain the following expressions:

$$\begin{aligned}
\Delta u_1(k) &= b(2\omega_1(k+1)\Delta\omega_1(k) - \Delta\omega_1^2(k) + 2\omega_2(k+1)\Delta\omega_2(k) - \Delta\omega_2^2(k) + \\
&\quad 2\omega_3(k+1)\Delta\omega_3(k) - \Delta\omega_3^2(k) + 2\omega_4(k+1)\Delta\omega_4(k) - \Delta\omega_4^2(k)) \\
\Delta u_2(k) &= b(2\omega_4(k+1)\Delta\omega_4(k) - \Delta\omega_4^2(k) - (2\omega_2(k+1)\Delta\omega_2(k) - \Delta\omega_2^2(k))) \\
\Delta u_3(k) &= b(2\omega_1(k+1)\Delta\omega_1(k) - \Delta\omega_1^2(k) - (2\omega_3(k+1)\Delta\omega_3(k) - \Delta\omega_3^2(k))) \\
\Delta u_4(k) &= d(-(2\omega_1(k+1)\Delta\omega_1(k) - \Delta\omega_1^2(k)) + (2\omega_2(k+1)\Delta\omega_2(k) - \Delta\omega_2^2(k)) - \\
&\quad (2\omega_3(k+1)\Delta\omega_3(k) - \Delta\omega_3^2(k)) + (2\omega_4(k+1)\Delta\omega_4(k) - \Delta\omega_4^2(k)))
\end{aligned} \tag{3.26}$$

From this, and knowing from the previous section that $\Delta\omega_i \in [\Delta\underline{\omega} \;\; \Delta\overline{\omega}] = [-60 \;\; 60]$ as well as that the bounds of $\omega_i(k+1)$ are shifted depending on the value of $\Delta\omega$, such that:

$$\omega_i(k+1) \in [\underline{\omega} \;\; \overline{\omega}] = \begin{cases} [0 \;\; 240] & \text{if} \quad \Delta\omega_i(k) = \Delta\underline{\omega} \\ [60 \;\; 320] & \text{if} \quad \Delta\omega_i(k) = \Delta\overline{\omega} \end{cases} \tag{3.27}$$

we define the minimum and maximum values of Δu_i as:

$$\begin{aligned} \Delta u_1 &\in [-4.3570 \;\; 4.3570] \\ \Delta u_2 &\in [-1.6526 \;\; 1.6526] \\ \Delta u_3 &\in [-1.6526 \;\; 1.6526] \\ \Delta u_4 &\in [-0.1008 \;\; 0.1008] \end{aligned} \tag{3.28}$$

3.3 Path planning

At a certain point of the simulations, it is desired to generate a path in a 3D map in order to perform path tracking and evaluate the performance of the controller. The reason is that the MPC might have a different performance for a certain prediction horizon depending on how complex the reference for the states is.

Therefore, by providing a reference extracted from a generated path on a map, we can provide the NMPC a reference complex enough so that we can test the performance in a situation similar to a real application, and not only in front of simple step inputs.

For that reason, it will be implemented a path finding algorithm using the technique of Probabilistic Roadmap Method (PRM).

The PRM is a relatively simple path planning algorithm that solves the problem of finding the shortest path between a starting point and a goal in a certain n-dimensional space, while avoiding obstacles.

The basic algorithm of a PRM is stated as follows:

Algorithm 1 PRM

Result: E
$V := 0$ $E := 0$ **while** $|V| < s$ **do**
  $q :=$ random sample in C **if** $q \in C_{free}$ **then**
    $V := V \cup q$
  end
end
forall $q \in V$ **do**
  $N :=$ neighbours of q chosen from V at an euclidean distance lower than d **forall** $n \in N$ **do**
    if $((q,n) \notin E) \wedge \Delta(q,n)$ **then**
      $E := E \cup (q,n)$
    end
  end
end

In Algorithm 1, V is the set of vertices, E the set of edges, C is the configuration space, C_{free} is the free configuration space (those points that are not inside an obstacle), (c,n) is the edge formed by the vertices q and n. $\Delta(q,n) \in \{0,1\}$ is a function that gives as output whether the two vertices are collision free or not.

This collision detection is carried out by tracing a straight line between the two configurations and checking for each discrete point of the line to be in the free space.

Algorithm 1 does actually give as output a set of edges representing the constructed roadmap. After that, however, an algorithm for computing the shortest path in that roadmap has to be applied. In this case, the Dijkstra's algorithm has been chosen due to it is implemented in a MATLAB function.

The Dijkstra algorithm needs as inputs the edges of the roadmap, as well as the weight of said edges (what would be the cost of going from one node to another). Therefore, the weights are defined simply as the Euclidean distance between each node, which seems logical given the fact that, as the space in which a quadrotor moves is the air, a priori there is no difference in the cost of traveling by two edges of the same distance

$$W(q_1, q_2) = \|q_2 - q_1\|_3 \tag{3.29}$$

For the simulations it is going to be used the 3D map shown in Figure 3.5. The black dots are considered obstacles by the path planner, and, as it can be seen, they form an structure that the path planner will have to sort. To avoid the planner from simply computing a path

that flies over the structure, samples will be only taken in the range of the z-axis going from 0 to 10 m (the chosen height for the structure).

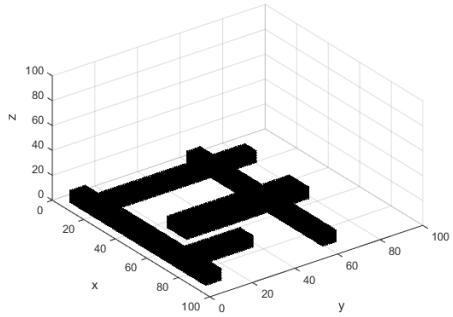

Figure 3.5: Perspective view of the map

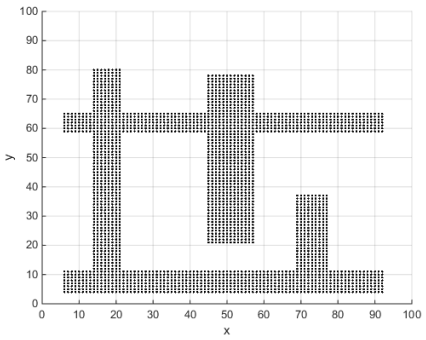

Figure 3.6: XY projection of the map

The map has been constructed to obtain a relatively realistic size for a path tracking. Being the sampling resolution of the map 1 m, it can be seen that the structure covers roughly for 90x80x10 m, which is an acceptable size for the case of considering it to be the 3D map of a building.

Since the configuration space of the quadrotor will only consider the XYZ degrees of freedom, and not the rotational ones (it will be considered that a collision does only depend on the position, and not the orientation), then the configuration space of the quadrotor coincides with the constructed map, with the structure in black as obstacle and the rest as free space.

Therefore, the PRM is ready for the path planning. As a demonstration of its performance, a path will be computed between the following points, more or less going from the outside of the building to the interior plaza:

$$S = (30\ 90\ 1) \tag{3.30}$$
$$G = (30\ 50\ 5)$$

with S being the starting point and G the goal.

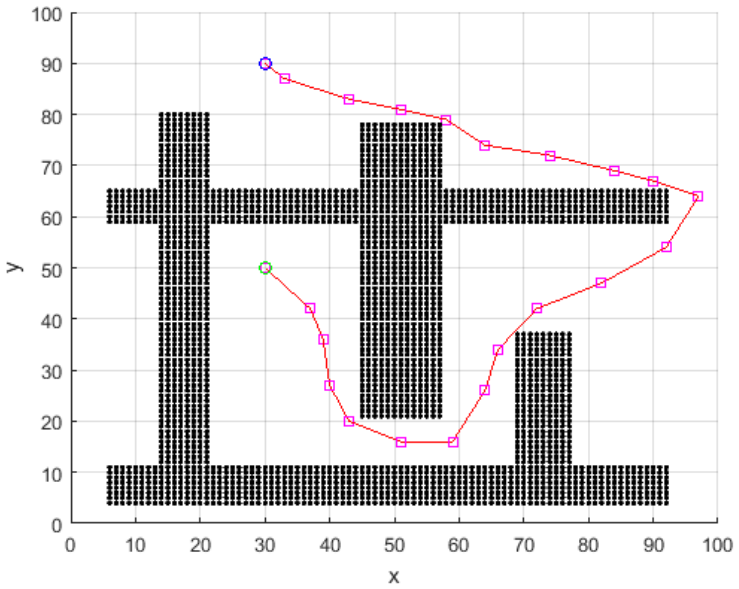

Figure 3.7: Computed path from (30, 90, 1) to (30, 50, 5) with s = 500 and d = 10. XY projection view. Blue circle: start point. Green circle: goal. Purple squares: samples in V. Red line: continuous path

Below (figure 3.8 and 3.9) the effects of varying both the number of samples s and the threshold distance d, with respect to those values of figure 3.7, can be seen.

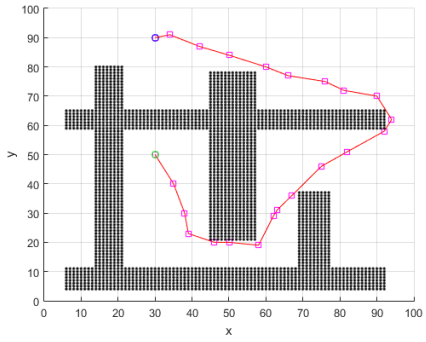

Figure 3.8: Path obtained with s = 1000 and d = 10

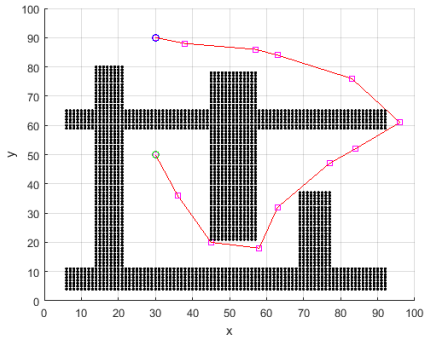

Figure 3.9: Path obtained with s = 500 and d = 20

The effect of reducing the number of samples is the obtention of a more optimal path than using a lower number of samples. Additionally, it increases the probability of finding a path connecting the starting point and the goal. However, it also increases the computational

cost.

This is also the case of the distance threshold d, which affects the number of neighbours to a certain sample that will be chosen to attempt a connection. Increasing it, the density of connections for a certain number of samples will increase, therefore increasing the possibilities of finding an optimal path.

As a general rule, a minimum number of samples are needed in order to have a minimum of probability that a path between the goal and the objective exists. Then, in addition, a suitable distance threshold has to be selected in order to be able to connect the samples. Generally speaking, some trade-off exists at low values of either of s or d. For relatively low values of s, high values of d have to be selected in order to connect the samples, and viceversa.

Now, as one might appreciate, this could also have an effect regarding the control of the quadrotor. Particularly, the selection of values of s and d such that a relatively sharp path is obtained might be detrimental in the performance of the controller, or even stability, if the region of attraction of the reference turns out to be small, which may require to increase the prediction horizon.

Therefore, if the controler shows strong stability properties and good performance, the form of the reference input should not be a factor to be considered as important for stability or performance purposes. However, this will be analyzed by simulation to properly test the actual effect of manipulating s and d in the necessary prediction horizon for the path tracking.

The following step is to interface the PRM with the NMPC controller.

The main issue is due to the fact that, while the NMPC controller works in the time domain, the path tracking is a spacial tracking, and not a temporal one. Therefore, the spacial references of the position have to be translated into temporal references so that it can be given as inputs to the controller. This will be done for each segment forming the path.

A velocity profile has then to be defined. Choosing a linear velocity profile of a certain value v, and knowing the sample time T_s and the distance that separates the i-th sample and the next (d_s), we can compute the number of discrete values that will form the reference signals (n_s), such as:

$$n_s = \frac{d_s}{T_s v} \tag{3.31}$$

Then, after computing n_s, we can linearly interpolate the values of the discrete points with which the position references for x, y and z will be defined, being this way ready to be passed

to the controller.

Chapter 4

Results

In this section we are going to show the results of of the control of the quadrotor via the NMPC schemes finally considered in Section 3.2, and its properties regarding stability, performance, feasibility and computational time will be compared.

The simulations will first show the results during the the process of tuning the controller consisting in:

1. Select relatively close starting point and goal in the map in order to reduce the simulation time, but providing still a reference that involves a 3-dimensional movement.

2. Initially consider unitary weight for the position states in P, $Q = 0_{4x4}$ and $R = 0_{4x4}$. Take as prediction horizon $N = 10$ and as sampling time $T_s = 0.1$ s.

3. In case the problem is infeasible, unstable or shows bad performance, some weighs will be added to Q and R.

4. After this, individual weights will be tuned to improve as much as possible the tracking, coupled with the increase on the prediction horizon, if needed.

Therefore, it is going to be considered, initially, a path going from starting point $(30,\ 40,\ 1)$ to ending point $(40,\ 50,\ 1)$, as shown in Figure 4.1.

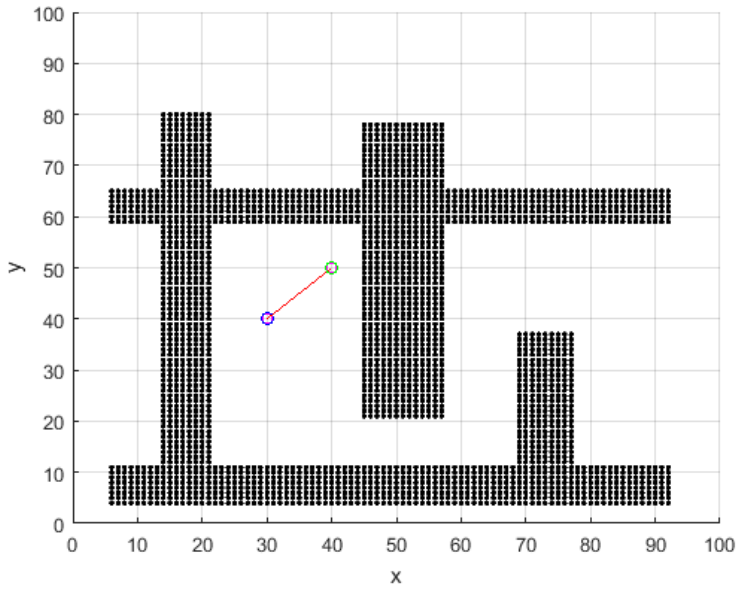

Figure 4.1: Computed path from (30, 40, 1) to (40, 50, 5). XY projection view. Blue circle: start point. Green circle: goal. Red line: continuous path

All simulations have been performed in Matlab, using the Yalmip toolbox to model the OCP, and using OptiToolbox (a free toolbox with several optimization solvers), from which Ipopt solver (a solver using Interior Point merhod) will be used for the optimization process.

The task will be performed using the CPU Intel Core I5-4590 @ 3.30 GHz.

The previous information is given in order to put into perspective the computational time that it may take to perform the simulations that will be shown.

We will show a table containing the maximum error, the mean error and the standard deviation of the error for each state with respect to its reference, with the aim of providing some sort of measure for the performance, as we could relate the maximum error to the overshoot, the mean error to the steady state error, and the standard deviation with the oscillations.

4.1 Initial tuning

The first simulation, once implemented the NMPC algorithm in Matlab, is done by using the following parameters:

$$\begin{aligned} T_s &= 0.1 \text{ s} & p_7 &= 1 \\ v &= 1 \ \frac{\text{m}}{\text{s}} & p_9 &= 1 \\ N &= 10 & p_{11} &= 1 \\ & & Q &= 0_{4x4} \\ & & R &= 0_{4x4} \end{aligned} \tag{4.1}$$

where T_s is the sampling time and v is the linear velocity reference for the quadrotor used in (3.31). Note that for the weights of the states P it has been used the notation p_i to refer to the individual weight of state x_i. This is done to avoid from having to introduce the whole matrix. From now on, if only some elements in P, Q or R are defined, the rest of them are assumed to be 0.

The result is that tuning the NMPC without weights in Q and R yields some problems in the computation of the optimization. Particularly, the optimizer seems to be unable to find a solution at the very first step of the simulation, even after several tries.

Therefore, we proceed to add some weight to u and Δu. The first choice is to define $Q = I_4$x4 and $R = I_4$x4, with I being the identity matrix. The parameters used for the following simulation are:

$$\begin{aligned} T_s &= 0.1 \text{ s} & p_7 &= 1 \\ v &= 1 \ \frac{\text{m}}{\text{s}} & p_9 &= 1 \\ N &= 10 & p_{11} &= 1 \\ & & Q &= I_{4x4} \\ & & R &= I_{4x4} \end{aligned} \tag{4.2}$$

However, this simulation is unable to finish, as at a 2%, the optimizer stops due to numerical problems.

Therefore, we will try reducing the weights on Q and R, such that:

$$
\begin{aligned}
T_s &= 0.1 \text{ s} & p_7 &= 1 \\
v &= 1 \ \frac{\text{m}}{\text{s}} & p_9 &= 1 \\
N &= 10 & p_{11} &= 1 \\
& & Q &= 0.1 I_{4x4} \\
& & R &= 0.1 I_{4x4}
\end{aligned} \tag{4.3}
$$

Again, the optimizer reports numerical problems. This time at the 6.7% of the simulation. The computation time up to this 6.7% of the simulation was of 153 seconds.

It can be seen how, up to the point in which the problem stopped, the states appeared to track relatively well the references (Figure 4.2) even though, the state z had an initial deviation.

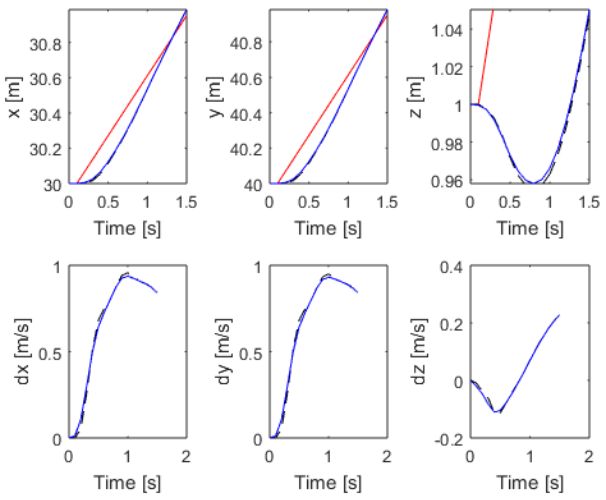

Figure 4.2: From left to right, and top to bottom, evolution of x, y, z, $\dot{x}$, $\dot{y}$ and $\dot{z}$. Red line: reference values. Black dotted line: discrete computed values by the optimizer. Blue line: continuous real values

A second try was given to the simulation with $N = 11$. This time it finished with a computational time $T_c = 1571$ seconds for a simulation time of 24.8 seconds. The obtained results show what seems to be good tracking of the references (Figure 4.3, except for the notable steady state error in z. Some measurements for the error in the position states with respect

to its references $\varepsilon_i = x_i - x_i^{ref}$ are given in Table 4.1.

As a side on, from now on we will consider the scalar r_T to represent the ratio between the computation time and the simulation ideal time. Thus, in this simulation $r_T = \frac{1571}{24.8} = 63.38$

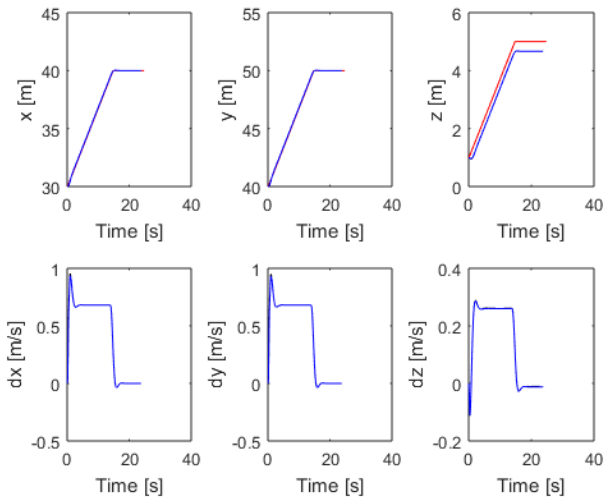

Figure 4.3: From left to right, and top to bottom, evolution of x, y, z, $\dot{x}$, $\dot{y}$ and $\dot{z}$. Red line: reference values. Black dotted line: discrete computed values by the optimizer. Blue line: continuous real values

for error ε_i:	max absolute value (max $\lvert\varepsilon\rvert$)	mean value	standard deviation
x [m]	0.2161	-0.0086	0.0387
y [m]	0.2166	-0.0087	0.0389
z [m]	0.3719	-0.3289	0.0438

Table 4.1: Data measurements of the error signals for the simulation using parameters in (4.3)

However, at a closer look to the pitch, roll and yaw derivatives (Figure 4.4), it presents initial oscillations before stabilizing.

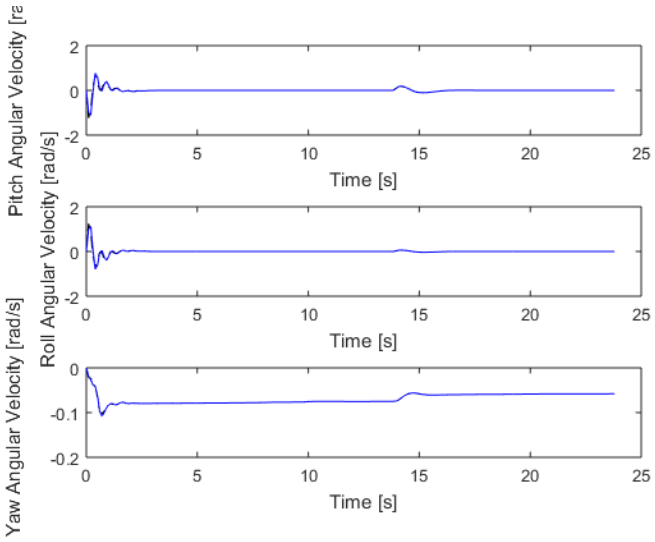

Figure 4.4: From top to bottom, evolution of $\dot{\phi}$, $\dot{\theta}$ and $\dot{\psi}$ for the simulation using parameters 4.3. Black dotted line: discrete computed values by the optimizer. Blue line: continuous real values

This may explain the previous issues, given that the initial report about numerical problems might be caused because of the problem being ill-conditioned, (this has not been proved, but it is a possibility) coupled with these derivative values. Then we will try to reduce as well the value of the derivatives of the pitch, roll, and yaw, with the aim of reducing the effect it might have in the numerical issues. Additionally, it might help reducing the relatively high computational time.

Therefore, assigning some weights to $\dot{\phi}$, $\dot{\theta}$ and $\dot{\psi}$, we simulate again with the following parameters:

$$
\begin{aligned}
T_s &= 0.1 \text{ s} & p_7 &= 1 \\
v &= 1 \ \frac{\text{m}}{\text{s}} & p_9 &= 1 \\
N &= 10 & p_{11} &= 1 \\
& & p_2 &= 0.1 \\
& & p_4 &= 0.1 \\
& & p_6 &= 0.1 \\
& & Q &= 0.1 I_{4x4} \\
& & R &= 0.1 I_{4x4}
\end{aligned}
\tag{4.4}
$$

where p_2, p_4 and p_6 are the weights for $\dot{\phi}$ (pitch derivative), $\dot{\theta}$ (roll derivative) and $\dot{\psi}$ (yaw derivative), respectively.

Indeed, the simulation using the parameters in (4.4) is able to finish, and after several trials it does not report numerical problems. In fact, the computational time is reduced dramatically, since now the simulation is completed with a computation time $T_c = 82$ seconds (down from 1571 seconds) for a simulation time of 24.8 seconds ($r_T = 3.3065$).

As for the results, in figures 4.5 and 4.6 the tracking of the reference position and the angle values are shown, respectively. From now on, simulation results will show only the position and angle states, in order to not overload the document with figures, unless it is necessary for the explanation to show other states, inputs, or, in general, any other plot.

Measurements on ε are shown in Table 4.2.

for error ε_i:	max absolute value (max $\lvert\varepsilon\rvert$)	mean value	standard deviation
x [m]	0.2510	-0.0091	0.0439
y [m]	0.2515	-0.0092	0.0440
z [m]	0.3716	-0.3289	0.0440

Table 4.2: Data measurements of the error signals for the simulation using parameters in (4.4)

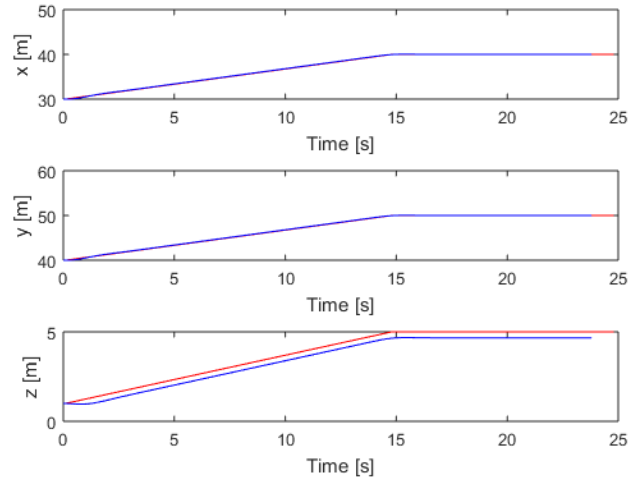

Figure 4.5: From and top to bottom, evolution of x, y and z for the simulation using parameters 4.4. Red line: reference values. Black dotted line: discrete computed values by the optimizer. Blue line: continuous real values

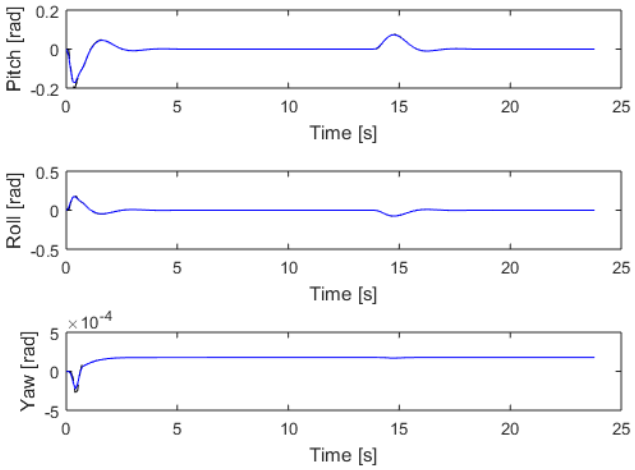

Figure 4.6: From top to bottom, evolution of ϕ, θ and ψ for the simulation using parameters 4.4. Black dotted line: discrete computed values by the optimizer. Blue line: continuous real values

Comparing Figures 4.5 and 4.3, as well as Tables 4.2 and 4.1, it can be appreciated the almost identical performance in the two simulations, with the obvious improvement made in this one by eliminating numerical issues and reducing drastically the computation time.

Still, it can be also appreciated the relatively high tracking error of state z. This is probably caused by the penalization in the control inputs. Therefore, we will try to reduce it by reducing the weighting values in Q

After some tests with different values of Q, it was found that, in fact, by setting $Q = 0_{4x4}$ (with parameters shown in (4.5)), the NMPC provides a considerably better performance, reducing almost entirely the steady state error in z. Results shown in Figures 4.7, 4.8 and Table 4.3.

$$
\begin{aligned}
T_s &= 0.1 \text{ s} & p_7 &= 1 \\
v &= 1 \ \frac{\text{m}}{\text{s}} & p_9 &= 1 \\
N &= 10 & p_{11} &= 1 \\
& & p_2 &= 0.1 \\
& & p_4 &= 0.1 \\
& & p_6 &= 0.1 \\
& & Q &= 0_{4x4} \\
& & R &= 0.1 I_{4x4}
\end{aligned}
\tag{4.5}
$$

for error ε_i:	max absolute value (max $\lvert\varepsilon\rvert$)	mean value	standard deviation
x [m]	0.2485	-0.0087	0.0444
y [m]	0.2489	-0.0087	0.0445
z [m]	0.0420	-0.0032	0.0065

Table 4.3: Data measurements of the error signals for the simulation using parameters in (4.5)

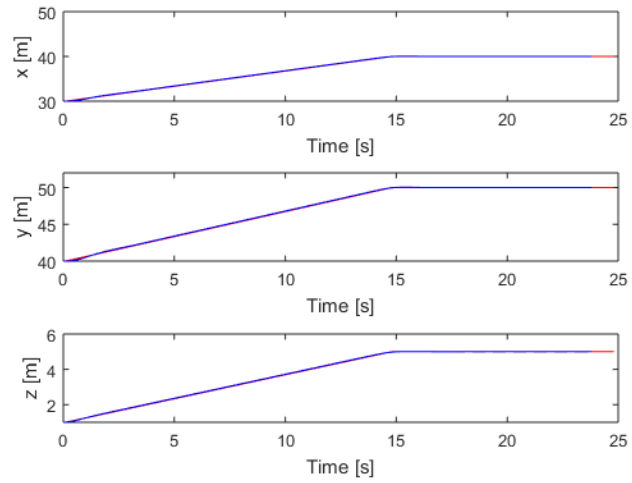

Figure 4.7: From and top to bottom, evolution of x, y and z for the simulation using parameters 4.5. Red line: reference values. Black dotted line: discrete computed values by the optimizer. Blue line: continuous real values

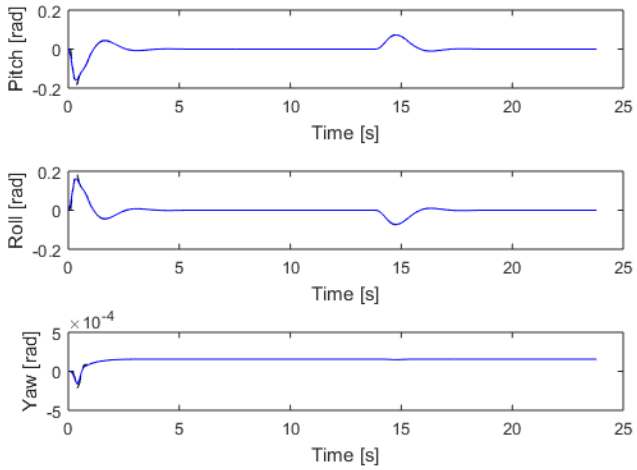

Figure 4.8: From top to bottom, evolution of ϕ, θ and ψ for the simulation using parameters 4.5. Black dotted line: discrete computed values by the optimizer. Blue line: continuous real values

At this point, seeing that we have obtained a relatively good tuning of the NPMPC, we may try now to reduce the computational time by reducing the prediction horizon N as much as possible while still guaranteeing stability and an acceptable performance.

Therefore, after some tests with $N < 10$, it was found that the minimum N that would meet with the required stability and performance is, in fact $N = 10$. For $N = 9$, the system, as shown in Figures 4.9 and 4.10 ($r_T = 3.9113$), becomes critically stable, therefore not meeting the performance required. For $N < 9$ the system becomes unstable, and at some point of the simulation the solver reports infeasibility.

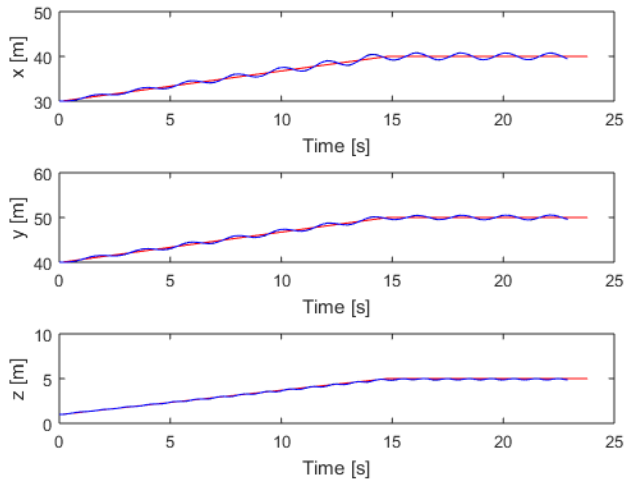

Figure 4.9: From and top to bottom, evolution of x, y and z for the simulation using parameters (4.5) with $N = 9$. Red line: reference values. Black dotted line: discrete computed values by the optimizer. Blue line: continuous real values

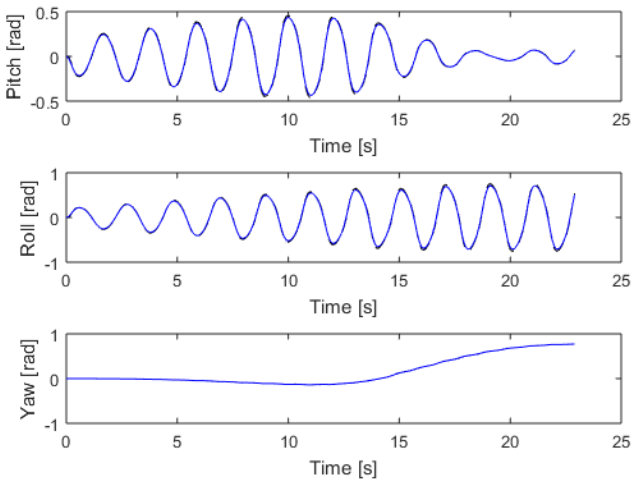

Figure 4.10: From top to bottom, evolution of ϕ, θ and ψ for the simulation using parameters (4.5) with $N = 9$. Black dotted line: discrete computed values by the optimizer. Blue line: continuous real values

At this point, in order to try to enforce stability for $N < 10$, we may try two different approaches:

1. Use of terminal weights

2. Use of terminal constraint set.

Since stability does not seem to be a problem for reasonable values of N, the motivation will be to obtain results that show similar performance with the one obtained with the initial scheme and parameters (4.5), with a prediction horizon N low enough that it compensates for the (probable) increase in the computational time T_c due to the addition of terminal weights or terminal constraints, obtaining therefore an effective reduction of the overall computational time of the problem.

4.1.1 Use of terminal weights

We will start by adding terminal weights to the finite horizon cost function $J(x, u)$, such that the optimization problem becomes:

$$J(x,u(\cdot)) = \sum_{k=0}^{N-2}((x(k)-x^{ref}(n+k))^T P(x(k)-x^{ref}(n+k)) + u(k)^T Q u(k) \tag{4.6}$$
$$+\Delta u(k)^T R \Delta u(k)) + (x(N-1)-x^{ref}(n+N-1))^T W P(x(N-1)-x^{ref}(n+N-1))$$
$$+u(N-1)^T Q u(N-1) + \Delta u(N)^T R \Delta u(N)$$

with W being a matrix such that:

$$W_{i,j} = \begin{cases} 0 & \forall \ \ i,j \mid (i \neq j) \cup (i = j \neq \{7,9,11\}) \\ \\ \lambda & \text{otherwise} \end{cases} \tag{4.7}$$

with λ being a scalar. This results in a weighting for the states z, x, y equal to λp_7, λp_9, λp_{11}, respectively.

However, it has to be taken into account that increasing excessively the value of the terminal weight may lead to the simulation requiring excessive computation time.

As an example, after performing several simulations with different values of λ for $N = 10$, we can see the effect of increasing λ to the computation time in Figure 4.11.

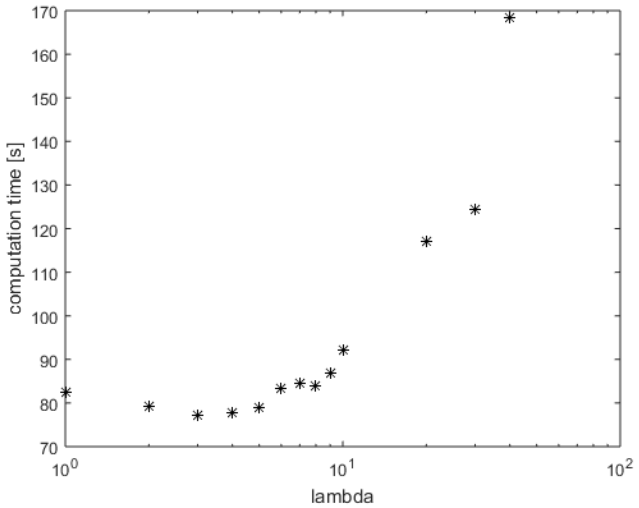

Figure 4.11: Evolution of the computation time with respect to the value of λ for $N = 10$. Semilogarthmic scale in x axis.

It can be seen how increasing a bit λ leads to obtain a lower computation time required to

perform the simulation, with a minimum at $\lambda = 3$ corresponding to $T_c = 77.23$ s ($r_T = 3.1141$). After that, it shows a tendency to increase the computational time as the value of λ increases, to the point in which, for $\lambda = 50$, the solver reports numerical problems at some point of the simulation and therefore it is unable to finish.

This behaviour is also appreciated for $N = 9$ in Figure 4.12. In this case, the minimum computation time is shifted towards a higher terminal weight $\lambda = 8$, corresponding to $T_c = 68.47$ s ($r_T = 2.7609$).

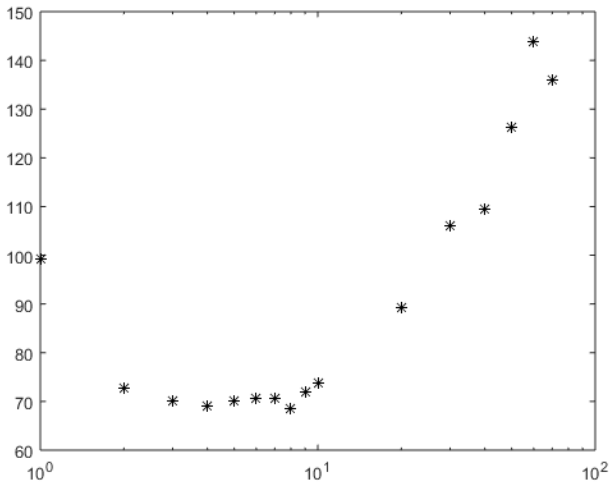

Figure 4.12: Evolution of the computation time with respect to the value of λ for $N = 9$. Semilogarthmic scale in x axis.

However, it has to be noted that the small differences seen between consecutive values of λ close to the minimum make it impossible to confirm which of them yield the actual lower T_c (given that some variations in computation time might be caused by external factors, such as the microprocessor having other tasks of outside the simulation). It is clear nonetheless that a tendency towards higher computation time is shown for relatively high λ values.

This is usefult to know, as it allows to determine the optimal λ value by increasing up until it is detected that the computation time starts increasing as well, although even this way, the process is quite a "trial and error" one.

Therefore, continuing with the aim of minimizing the prediction horizon N to reduce as well the computation time, after some more simulations we obtain the lowest computation time at $N = 8$ and $\lambda = 12$, with $r_T = 2.5645$ and the performance shown in Figures 4.13,

4.14 and Table 4.4. With $N < 8$, either higher r_T are obtained, or the problem is directly infeasible for apparently any λ.

The simulation parameters, in this case, are the following:

$$\begin{aligned} T_s &= 0.1 \text{ s} & p_7 &= 1 \\ v &= 1 \ \frac{\text{m}}{\text{s}} & p_9 &= 1 \\ N &= 9 & p_{11} &= 1 \\ \lambda &= 12 & p_2 &= 0.1 \\ & & p_4 &= 0.1 \\ & & p_6 &= 0.1 \\ & & Q &= 0_{4\text{x}4} \\ & & R &= 0.1 I_{4\text{x}4} \end{aligned} \tag{4.8}$$

for error ε_i:	max absolute value (max $\lvert\varepsilon\rvert$)	mean value	standard deviation
x [m]	0.1732	-0.0045	0.0245
y [m]	0.1743	-0.0046	0.0246
z [m]	0.0538	0.0017	0.0103

Table 4.4: Data measurements of the error signals for the simulation using parameters in (4.8)

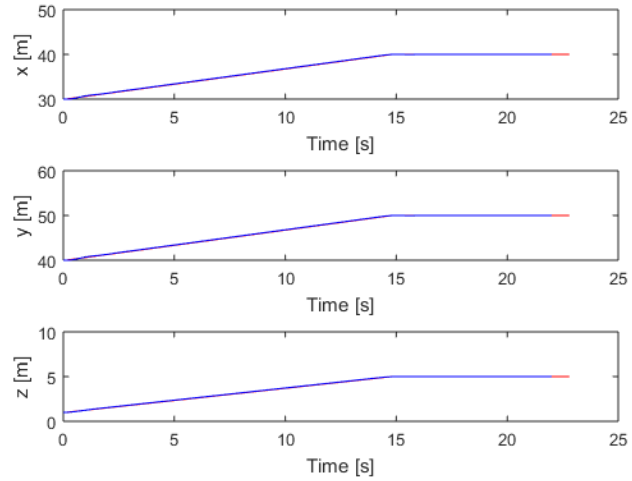

Figure 4.13: From and top to bottom, evolution of x, y and z for the simulation using parameters 4.8. Red line: reference values. Black dotted line: discrete computed values by the optimizer. Blue line: continuous real values

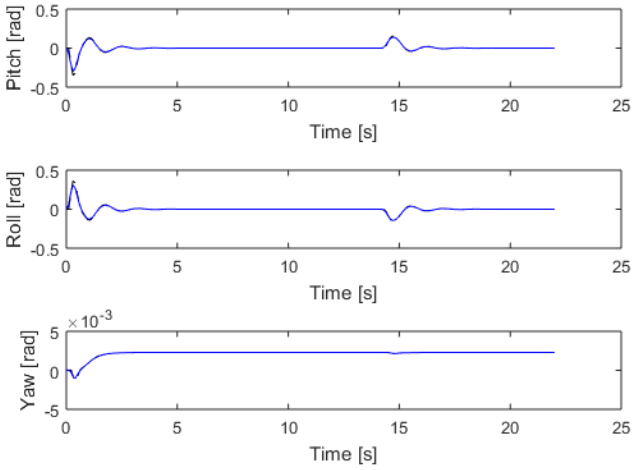

Figure 4.14: From top to bottom, evolution of ϕ, θ and ψ for the simulation using parameters 4.8. Black dotted line: discrete computed values by the optimizer. Blue line: continuous real values

Performance can be considered quite good, displaying a considerably accurate tracking, nearly identical to that of previous simulations.

4.1.2 Use of terminal constraints

At this point, in which the use of terminal weights does not provide more improvements for $N < 8$, we can now select the scheme of the NMPC using endpoint terminal constraints to try reduce again N and, possibly, r_T.

In the use of the previous scheme, the use of terminal weight λ not only allowed to stabilize the system for a reduced prediction horizon, but also, up to a certain value, even was able to reduce computation time for an already stable problem without terminal weights. However, in this case we cannot expect to for a fixed N reduce r_T with the introduction of terminal constraints, given that the optimization of a problem with additional constraints will inevitably carry a higher numerical effort.

For that reason, the simulations performed with this scheme will aim to improve the computation time of the best obtained configuration of the NMPC scheme using terminal weights, which is using a prediction horizon $N = 8$.

We want to first assess whether there are significant improvements in the performance (such as completely reducing the steady state error) for $N = 8$. Therefore, the simulation using the scheme in (3.11) yields the results shown in Table 4.5 and Figures 4.15.

The parameters used for the simulation are the following:

$$
\begin{aligned}
T_s &= 0.1 \text{ s} & p_7 &= 1 \\
v &= 1 \ \frac{\text{m}}{\text{s}} & p_9 &= 1 \\
N &= 8 & p_{11} &= 1 \\
& & p_2 &= 0.1 \\
& & p_4 &= 0.1 \\
& & p_6 &= 0.1 \\
& & Q &= 0 I_{4\text{x}4} \\
& & R &= 0.1 I_{4\text{x}4}
\end{aligned}
\tag{4.9}
$$

for error ε_i:	max absolute value (max $\|\varepsilon\|$)	mean value	standard deviation
x [m]	0.1566	-0.0063	0.0256
y [m]	0.1598	-0.0065	0.0261
z [m]	0.0505	-0.0012	0.0060

Table 4.5: Data measurements of the error signals for the simulation using parameters in (4.9)

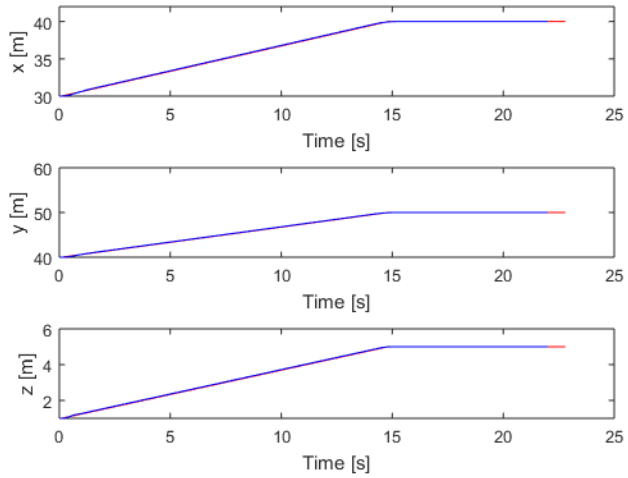

Figure 4.15: From and top to bottom, evolution of x, y and z for the simulation using parameters 4.9. Red line: reference values. Black dotted line: discrete computed values by the optimizer. Blue line: continuous real values

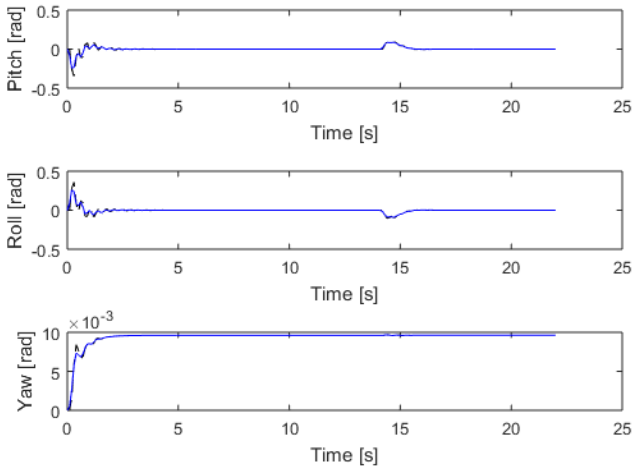

Figure 4.16: From top to bottom, evolution of ϕ, θ and ψ for the simulation using parameters 4.9. Black dotted line: discrete computed values by the optimizer. Blue line: continuous real values

The simulation is completed in 140 seconds, which is a big increase with respect to the 63 seconds of the scheme with terminal weights (meaning that $r_T = 5.6452$. Then, we can see that the performance is almost identical to that of the scheme using terminal weights for $N = 8$.

So finally, we proceed to simulate for $N < 8$ to try to reduce r_T.

However, for such prediction horizon values, the optimizer is unable to find a feasible solution, which therefore renders the use of terminal constraints, for our problem, a worse choice than the use of terminal weights.

4.2 Testing for different linear velocities

The linear velocity reference of the quadrotor, although it is not set explicitly into the state constraint set, it implicitly exists because of the value v used to transform the spatial domain references into time domain references.

So far, a $v = 1$ has been used. Therefore, we desire to determine the maximum velocity for which the quadrotor is able to correctly track the reference position.

Using the NMPC with terminal weights with parameters (4.8), we have obtained the maximum velocity at around $v = 2.5$, for which the results are shown in Figures 4.17, 4.18 and

Table 4.6. The simulation is completed with $r_T = 3.2207$.

for error ε_i:	max absolute value (max $\|\varepsilon\|$)	mean value	standard deviation
x [m]	0.4824	-0.0685	0.0983
y [m]	0.4977	-0.0699	0.1024
z [m]	0.1770	-0.0355	0.0430

Table 4.6: Data measurements of the error signals for the simulation using parameters in (4.8) with $v = 2.5$

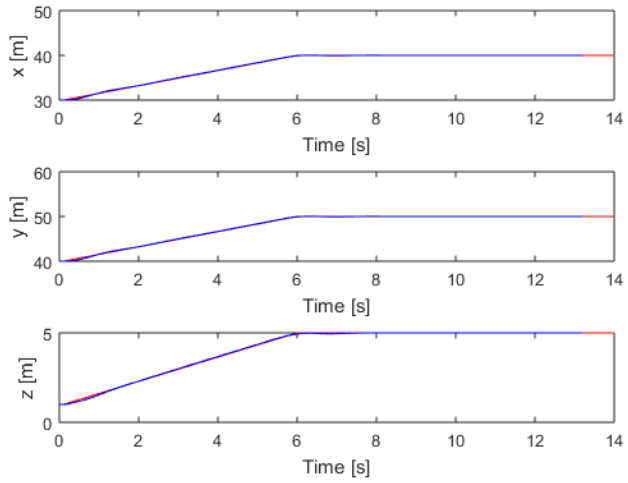

Figure 4.17: From and top to bottom, evolution of x, y and z using parameters in 4.8 with $v = 2.5$. Red line: reference values. Black dotted line: discrete computed values by the optimizer. Blue line: continuous real values

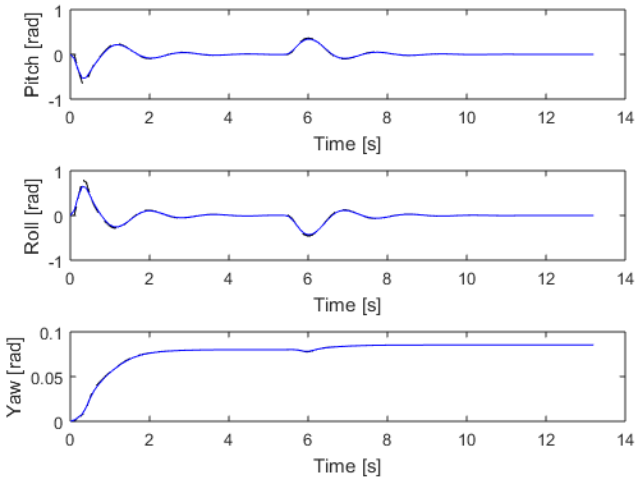

Figure 4.18: From top to bottom, evolution of ϕ, θ and ψ using parameters in 4.8 with $v = 2.5$. Black dotted line: discrete computed values by the optimizer. Blue line: continuous real values

It can be appreciated an almost equally good performance to that of previous simulations, with a slightly higher oscillation in z. For $v > 2.5$, the problem becomes infeasible. A priori, seeing that for $v = 2.5$ the system is stable and shows good performance, one would not expect the problem to become infeasible without showing signs of instablity for close values. However, the issue, in this case, does not come from instability, but actually because of a violation of the constraint set in the optimal regulator of the motors. The increased velocity reference makes the quadrotor to need higher ω_i on the motors, therefore arriving to a point in which, due to operational limitations, they are not able to provide it.

For that reason, in order to increase the velocity of the quadrotor, at this point it could be interesting to put again some weight in Q, which would require also to add additional weight to p_7 (state z) because of the steady state eror it would introduce, or reduce the upper bounds in $\mathbb{U}$ by some factor β.

For example, if we consider a matrix $Q = 0.1I_{4x4}$, as well as $p_7 = 2$, we are able to increase the velocity up to $v = 3.5$, with $r_T 03.0846$.

Thus, the simulation yields the results shown in table 4.7 and figures 4.19 and 4.20. Increasing oscilations with respect to simulations with lower v cna be appreciated through the value of the standard deviation of the errors, as well as through figure 4.20.

The parameters used for this simulation are the following:

$$
\begin{aligned}
T_s &= 0.1 \text{ s} & p_7 &= 2 \\
v &= 3.5 \ \frac{\text{m}}{\text{s}} & p_9 &= 1 \\
N &= 8 & p_{11} &= 1 \\
\lambda &= 12 & p_2 &= 0.1 \\
& & p_4 &= 0.1 \\
& & p_6 &= 0.1 \\
& & Q &= 0.1 I_{4x4} \\
& & R &= 0.1 I_{4x4}
\end{aligned}
\tag{4.10}
$$

for error ε_i:	max absolute value (max $\lvert\varepsilon\rvert$)	mean value	standard deviation
x [m]	0.8312	-0.0848	0.2147
y [m]	0.8512	-0.0866	0.2194
z [m]	0.2287	-0.0872	0.0593

Table 4.7: Data measurements of the error signals for the simulation using parameters in (4.10)

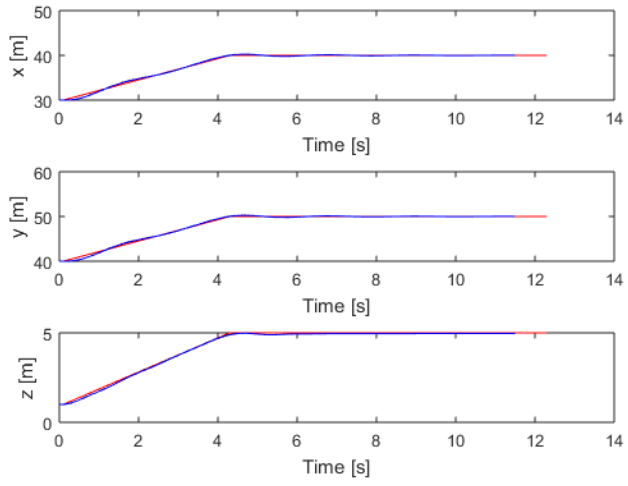

Figure 4.19: From and top to bottom, evolution of x, y and z using parameters in (4.10). Red line: reference values. Black dotted line: discrete computed values by the optimizer. Blue line: continuous real values

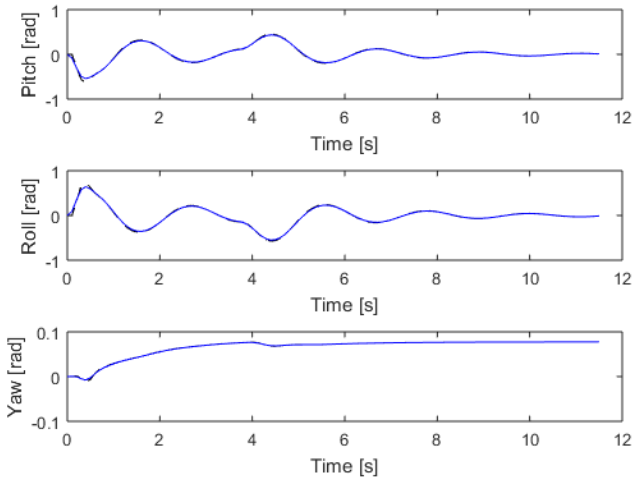

Figure 4.20: From top to bottom, evolution of ϕ, θ and ψ using parameters in (4.10). Black dotted line: discrete computed values by the optimizer. Blue line: continuous real values

If we were to reduce the upper bounds in $\mathbb{U}$ using again the parameters in (4.8) and a scaling factor $\beta = 0.7$ for the upper bounds, then the results would be the following (Table 4.8 and Figures 4.21, 4.22. Note that an excessive reduction of the upper bounds in $\mathbb{U}$ would mean that the controler might even lose the ability to maintain the quadcopter in air.

for error ε_i:	max absolute value (max $\lvert\varepsilon\rvert$)	mean value	standard deviation
x [m]	0.8507	-0.1098	0.1945
y [m]	0.8882	-0.1120	0.2036
z [m]	0.4352	-0.0650	0.1024

Table 4.8: Data measurements of the error signals for the simulation using parameters in (4.10) and $\beta = 0.7$

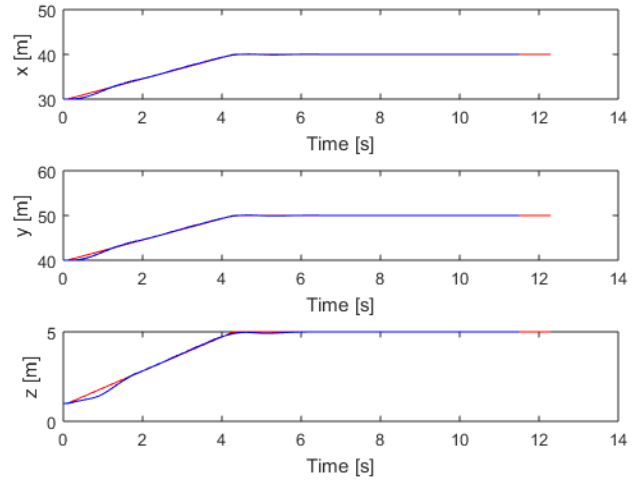

Figure 4.21: From and top to bottom, evolution of x, y and z using parameters in (4.8) and $\beta = 0.7$. Red line: reference values. Black dotted line: discrete computed values by the optimizer. Blue line: continuous real values

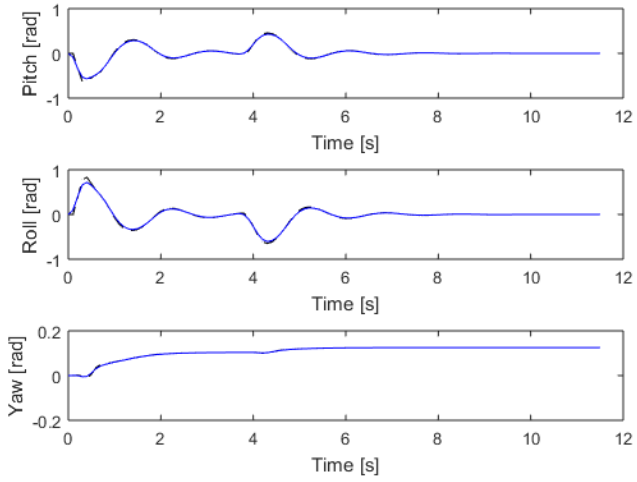

Figure 4.22: From top to bottom, evolution of ϕ, θ and ψ using parameters in (4.8) and $\beta = 0.7$. Black dotted line: discrete computed values by the optimizer. Blue line: continuous real values

The performance is showing, with respect to the use of weighting matrix Q, a sligthly larger maximum error, which can be seen at the image as a long overshoot, which leads to a similar standard deviation, even if there are less oscillations, as well as similar mean error. How-

ever, the problem is solved with a computation time of 58.17 s, in front of the 37.94 s of the other approach.

From this point, in general, for a fixed value of N, it is observed that increasing the values on Q, paired with suitable weight values of p_7 to reduce the steady state error, the maximum linear velocity reference v can be increased such as seen in Figure 4.23. Up to a point in which the increase in v becomes stagnant, it seemengly shows a linear relation with Q.

However, this increase in v has an effect in the performance, which is gradually lost, therefore arriving to a point in which the problem, due to oscillations with large overshoots, is increasingly difficult to keep being feasible (see Figures 4.24 and 4.25 for the response with $Q = 1 and v = 9.4$, completed with $r_T = 5.12$). Computational time is also graduallly increased.

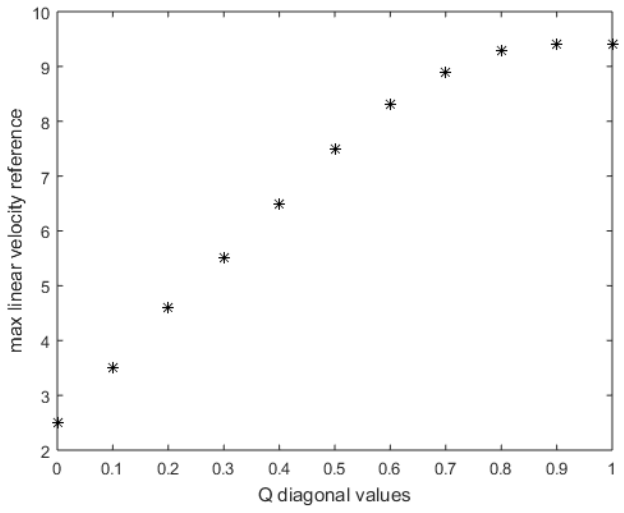

Figure 4.23: Maximum v in front of the diagonal values in Q for $N = 8$.

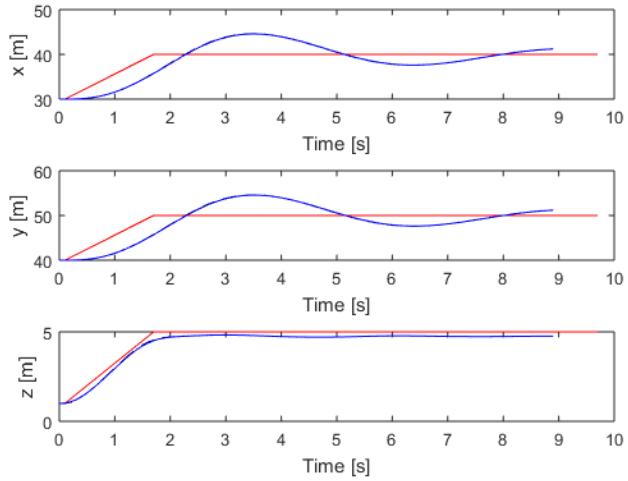

Figure 4.24: From and top to bottom, evolution of x, y and z using parameters in (4.8) with $Q = 1$ and $p_7 = 10$. Red line: reference values. Black dotted line: discrete computed values by the optimizer. Blue line: continuous real values

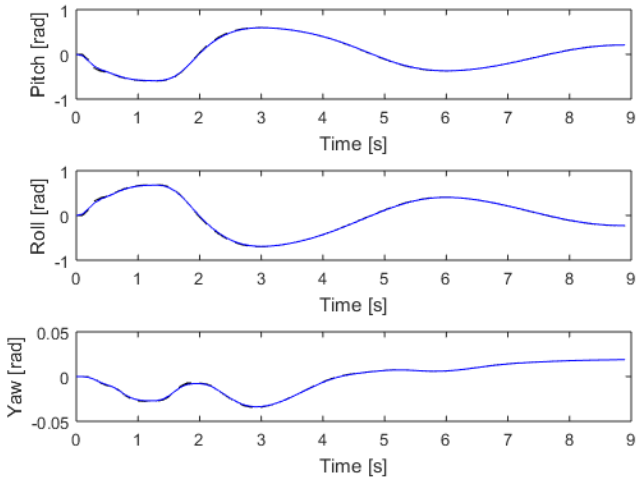

Figure 4.25: From top to bottom, evolution of ϕ, θ and ψ using parameters in (4.8) with $Q = 1$ and $p_7 = 10$. Black dotted line: discrete computed values by the optimizer. Blue line: continuous real values

Of course, the maximum value of v may also be increased by increasing N. However, note that the increase of N would probably imply a relatively high increase in computational time to solve the problem. This, coupled with the fact that simulations have shown that an

increase in N makes the maximum value of v increase very slighlty, makes it preferable to not vary the value of N in order to increase v (for instance, using N =9, 10, 11 leads to a max v = 2.6, 3, 3.3, respectively.

To finish this subsection, we may select as a suitable configuration to increase v one which does not reduce significantly the performance with respect to the simulations using (4.8).

Now, to decide what is a good performance in order to determine if a configuration does suit us or not, we may decide it mainly based on the value of the maximum error. Given that the problem is, at the end of the day, a path traking, through a map that may contain obstacles, it is undesired to have a maximum error with resspsect to the path such that it might imply a collision.

So, having in this case a map with a resolution of 1 m, that will be considered as the maximum admissible error. Therefore, this corresponds, in fact, to the first configuration tested for increasing v, which is (4.10).

4.3 Complete path tracking problem

Up until now, all simulations have been done using a simple ramp reference for the three position states in order to reduce the consumed time in performing the task of tuning the controller for certain situations.

For that reason, we now want to test stability and performance in the situation of tracking a complete path in the map.

To begin with, we will use the scheme and configuration selected as the best choice in Section4.1, which is the use of terminal weights with parameters in (4.8).

The first simulation will involve the tracking of the path going from point (30, 90, 1) to (30, 50, 5). The path is generated using 500 random samples and a distance threshold $d = 10$, and is shown in Figure 4.26. The obtained results are shown in Table 4.9 and Figures 4.27 and 4.28.

It can be appreciated that, as expected from the previous results with a simpler reference, when tracking a more complex path the controller is able to do it with equally good performance. The tracking shows high accuracy, even in the sharp turns appearing in the path.

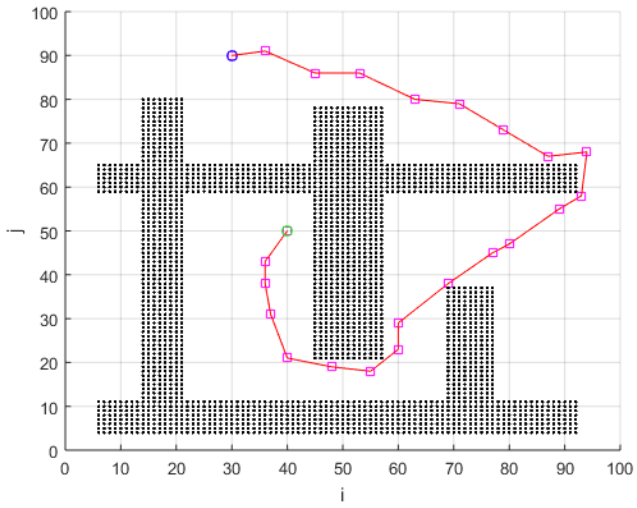

Figure 4.26: Computed path from (30, 90, 1) to (30, 50, 5) with s = 500 and d = 10. XY projection view. Blue circle: start point. Green circle: goal. Purple squares: samples in V. Red line: continuous path

for error ε_i:	max absolute value (max $\lvert\varepsilon\rvert$)	mean value	standard deviation
x [m]	0.2054	-0.0006	0.0117
y [m]	0.0695	-0.0001	0.0070
z [m]	0.1015	0.0041	0.0074

Table 4.9: Data measurements of the error signals for the simulation of the complete path tracking using parameters in (4.9)

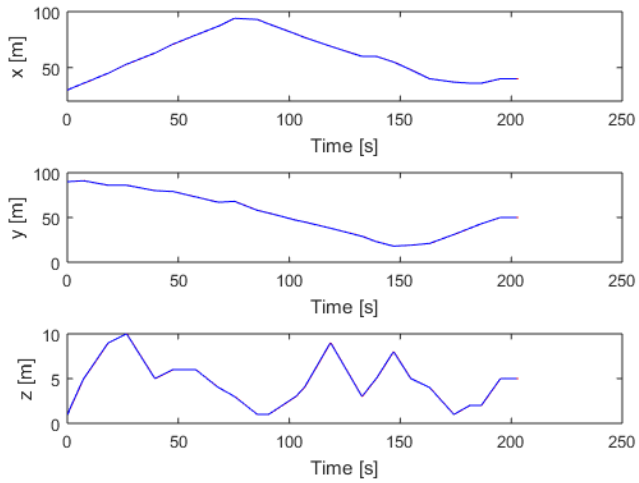

Figure 4.27: From and top to bottom, evolution of x, y and z for the simulation of the path tracking. Red line: reference values. Black dotted line: discrete computed values by the optimizer. Blue line: continuous real values

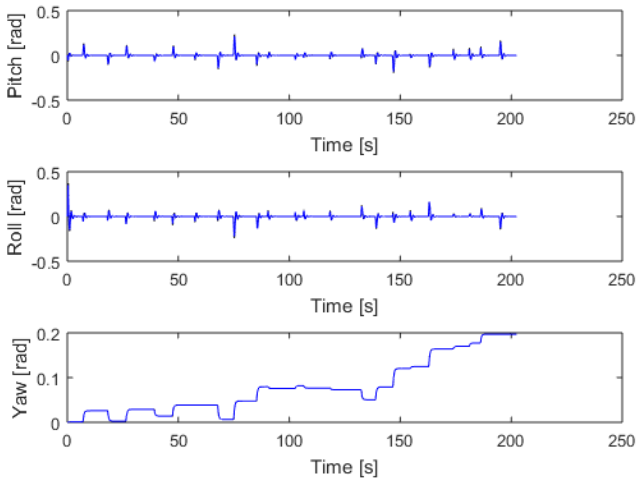

Figure 4.28: From top to bottom, evolution of ϕ, θ and ψ for the simulation of the path tracking. Black dotted line: discrete computed values by the optimizer. Blue line: continuous real values

If we consider the configuration in (4.10) for the maximum considered velocity, then the results, as well, meet with the ones seen for the ramp reference (see Figures 4.29 and 4.30 and Table 4.10). The problem is solved with $r_T = 3.7566$

for error ε_i:	max absolute value (max $\|\varepsilon\|$)	mean value	standard deviation
x [m]	1.1973	-0.0181	0.2347
y [m]	0.4881	0.0350	0.1712
z [m]	0.3474	-0.0558	0.0605

Table 4.10: Data measurements of the error signals for the simulation of the complete path tracking using parameters in (4.10)

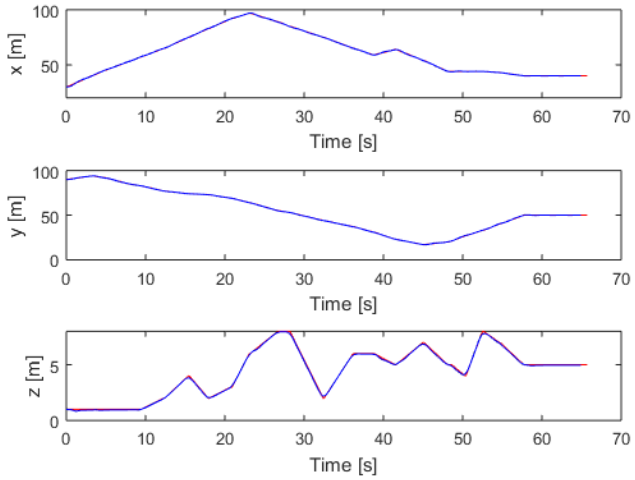

Figure 4.29: From and top to bottom, evolution of x, y and z using parameters in (4.8) with $Q = 1$ and $p_7 = 10$. Red line: reference values. Black dotted line: discrete computed values by the optimizer. Blue line: continuous real values

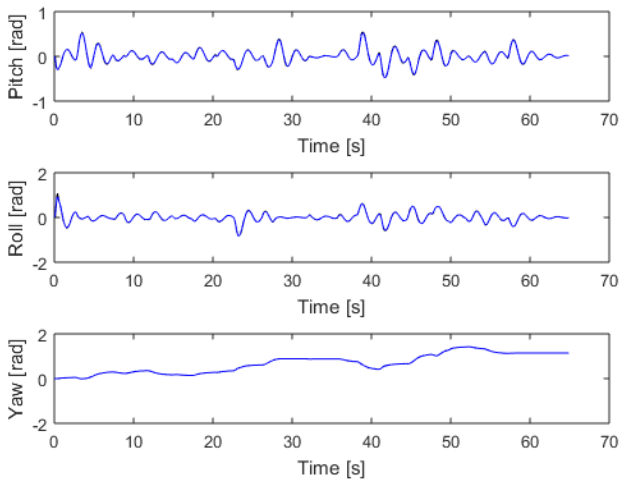

Figure 4.30: From top to bottom, evolution of ϕ, θ and ψ using parameters in (4.8) with $Q = 1$ and $p_7 = 10$. Black dotted line: discrete computed values by the optimizer. Blue line: continuous real values

However, even if it has a seemingly good performance, something has to be pointed out. Given that the map has a certain resolution (in this case, 1 m), then that is the minimum distance that may separate the computed shortest path from the obstacles. For that reason, a path tracking that shows in the error an euclidean distance higher than said resolution has the possibility of being actualy colliding with an obstacle at some point of the trajectory.

In this simulation, in fact, the maximum euclidean distance error during the trajectory tracking has been of 1.29 m. Seeing a close up image of the closer points of the path to the obstacles (Figure 4.31. Note that the map is different from the previous simulations, due to the random nature of the path planning algorithm), it can be appreciated that no collision occurs, although it does not imlpy at all that a collision could not happen for other reference paths.

For that reason might be advisable to reduce the maximum speed allowed in order to avoid collision, although the best solution would be to define a time-varying state constraint set.

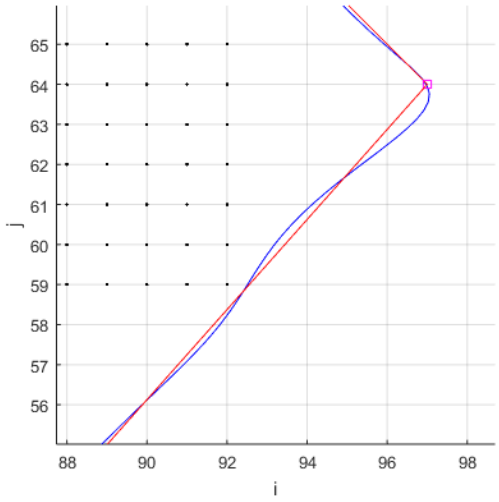

Figure 4.31: Close up look at the spatial trajectory of the quadrotor position (blue) and the path reference (red). Black dots: obstacles

4.4 Path tracking under modelling errors and disturbances

So far, all simulations have been done considering a ''real model" equal to the control model (that is, an ideal model for the simulation).
For the following simulations we are going to consider that the real model and the control model have some variations among themselves.
This is something that generally happens in a practical application. At some degree, there are dynamics that are either omitted in the control model because they do not provide enough increment in the performance with respect to the cost of implementing them (or computational cost), and there are also dynamics that are simply unknown. Therefore, the NMPC faces the problem of computing the inputs that optimize the state tracking, while the actual states of the real system may follow an evolution that is actually not the optimal. Additionally, control systems usually suffer from perturbations in the states and noise in the sensors for different reasons (in this case, a quadrotor faces disturbances from the wind). The ability of the controller to stabilize the system in front of those errors is called robustness.
To a certain degree in the errors between the control model and the real one, the NMPC should be able to provide some robustness in the control, and this is what we are going to test with this simulation.
The errors in the control model with respect to the simulated one will consist in:

- Addition of the propeler gyroscopic effect
- Addition of the hub moments
- Addition of the friction forces
- Addition of wind

The propeller gyroscopic effect (2.6) is easy to incorporate in the quadrotor model, since it depends on a known parameter (J_r) and the motor angular velocities.
Regarding the hub force, the equation that defines it((2.1.1)) shows that it, like the thrust force ((2.1)), is also dependant on the square of the rotor's angular speed ω. This allows us to consider the hub force as a parametric error in the thrust force.
Grouping the constants into a single coefficient, we have:

$$T_i = C_T \rho A (\Omega R_{rad})^2 = K_T \Omega_i^2$$
$$H_i = C_H \rho A (\Omega R_{rad})^2 = K_H \Omega_i^2$$

Now, to determine how much of a quantity is enough for the hub force to be neglected is a bit subjective, but we might consider a low value in the hub coefficient with respect to the thrust coefficient, in order to justify neglecting this effect in the control model. If we consider the Hub coefficient as a 5 % of that of the Thrust, then we have:

$$H_i = 0.05 K_T \Omega_i^2 \tag{4.11}$$

As for the friction force, it is an effect dependant on several factors. From (2.23) and (2.26), we can see that it depends on parameters such as the air density, the structure contact area, and the friction coefficient.
The friction coefficients C_x and C_y are assumed to be the same, because of symmetry in the quadrotor structure (for that reason, it will be referred from now on as C_f.
Normally, the friction coefficient would be obtained experimentally. The reasons for this is that, as seen in [6], the friction coefficient is highly dependant on the form of the object, the angle of incidence, and, most importantly, to the Reynolds number (R_e), to the point in which empirical expressions to compute C_f only holds for certain ranges of R_e. For that reason, it has not been possible to obtain a reliable approximated value for the friction coefficient. Therefore, the approach will be to perform some simulations in a range of values for

C_f with a maximum value such that the controller is able to provide stability and a certain performance.

Finally, the disturbances caused by the wind are added as disturbances in the velocity terms appearing in (2.23) and (2.26), obtaining that way:

$$m\ddot{y} = (-c\psi s\phi + s\psi s\theta c\phi)(\sum_{i=1}^{4} T_i) - \sum_{i=1}^{4} H_{yi} - \frac{1}{2} C_y A_c \rho (\dot{y} - \dot{y}_w)|\dot{y} - \dot{y}_w| \tag{4.12}$$
$$m\ddot{x} = (s\psi s\phi + c\psi s\theta c\phi)(\sum_{i=1}^{4} T_i) - \sum_{i=1}^{4} H_{xi} - \frac{1}{2} C_x A_c \rho (\dot{x} - \dot{x}_w)|\dot{x} - \dot{x}_w|$$

where $\dot{x}_w$ and $\dot{y}_w$ are the wind velocity components in the x-axis and y-axis, respectively.

Two wind profiles have been selected to test in simulation. One of them consists in a constant wind profile of 20.8 $\frac{\text{m}}{\text{s}}$, and the other consists in a sinusoidal wind profile of amplitude equal to 20.8 $\frac{\text{m}}{\text{s}}$

First, simulations are performed using a reference path such as the one in Figure 4.1 in order to reduce the time employed to perform the simulations. After obtaining the maximum values of C_f for this reference, we will test it on the complete path reference. Initially, the continuous wind disturbance will be considered, applied to the x-axis. So, x_w and y_w are defined as:

$$\dot{x}_w = 20.8 \tag{4.13}$$
$$\dot{y}_w = 0$$

The following controller parameters are used:

$$
\begin{aligned}
T_s &= 0.1 \text{ s} & p_7 &= 1 \\
v &= 2.4 \frac{\text{m}}{\text{s}} & p_9 &= 1 \\
N &= 8 & p_{11} &= 1 \\
\lambda &= 12 & p_2 &= 0.1 \\
& & p_4 &= 0.1 \\
& & p_6 &= 0.1 \\
& & Q &= 0_{4x4} \\
& & R &= 0.1 I_{4x4}
\end{aligned}
\tag{4.14}
$$

The maximum value of C_f for which the simulation remains feasible is found at $C_f = 520$. The results obtained can be seen at Table 4.11 and Figures 4.32 and 4.33. It shows an overall good performance, similar to simulations considering an ideal model for the quadrotor, although a slight increase in the steady state error in x, caused by the constant wind disturbance, can be observed. The simulation is completed with $r_T = 4.72$.

for error ε_i:	max absolute value (max $\lvert\varepsilon\rvert$)	mean value	standard deviation
x [m]	0.4384	0.0151	0.1095
y [m]	0.5029	-0.0840	0.1004
z [m]	0.1959	-0.0345	0.0457

Table 4.11: Data measurements of the error signals for the simulation of the complete path tracking applying continuous wind disturbance and using parameters (4.14)

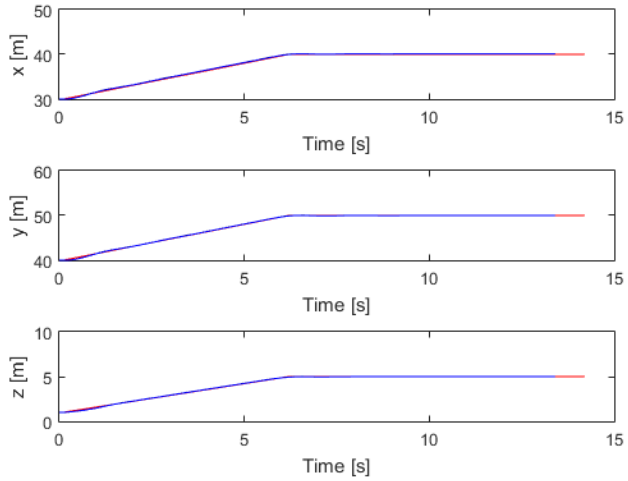

Figure 4.32: From and top to bottom, evolution of x, y and z applying continuous wind disturbance and using parameters (4.14). Red line: reference values. Black dotted line: discrete computed values by the optimizer. Blue line: continuous real values

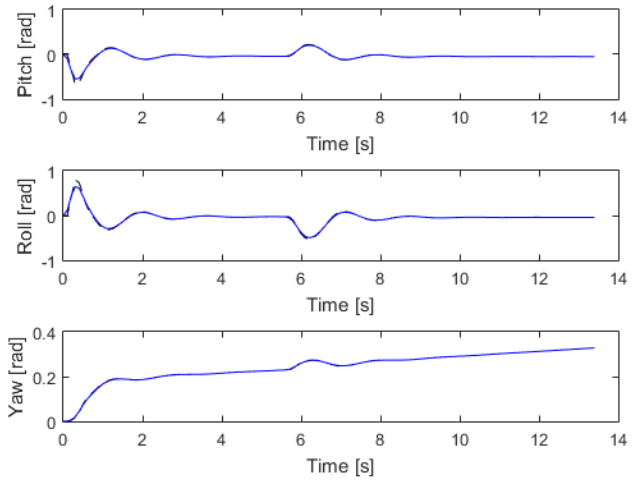

Figure 4.33: From top to bottom, evolution of ϕ, θ and ψ applying continuous wind disturbance and using parameters (4.14). Black dotted line: discrete computed values by the optimizer. Blue line: continuous real values

Increasing the prediction horizon to $N = 9$, the maximum friction coefficient allowed is, approximately, $C_f = 1490$, obtaining the results shown in Table 4.12 and Figures 4.34 and

4.35. The simulation is completed with $r_T = 5.7$. It can be observed that, even if it is able to keep stability, performance is reduced due to increasing C_f.

for error ε_i:	max absolute value (max $\|\varepsilon\|$)	mean value	standard deviation
x [m]	0.3779	0.2452	0.1672
y [m]	0.5542	-0.1089	0.1172
z [m]	0.2169	-0.0208	0.0521

Table 4.12: Data measurements of the error signals for the simulation of the complete path tracking applying continuous wind disturbance and using parameters (4.14) with $N = 9$

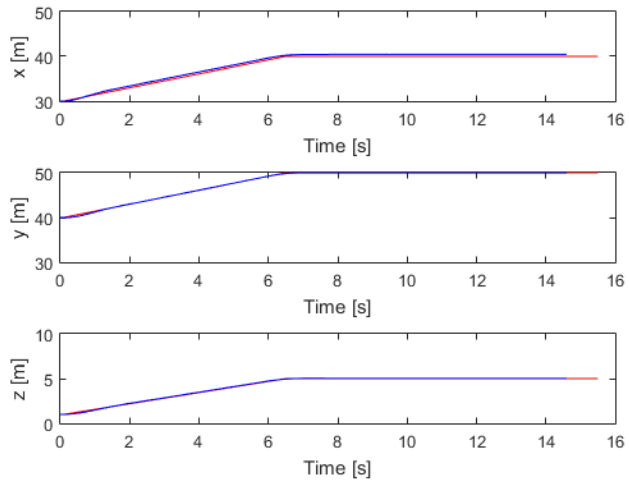

Figure 4.34: From and top to bottom, evolution of x, y and z applying continuous wind disturbance and using parameters (4.14) with $N = 9$. Red line: reference values. Black dotted line: discrete computed values by the optimizer. Blue line: continuous real values

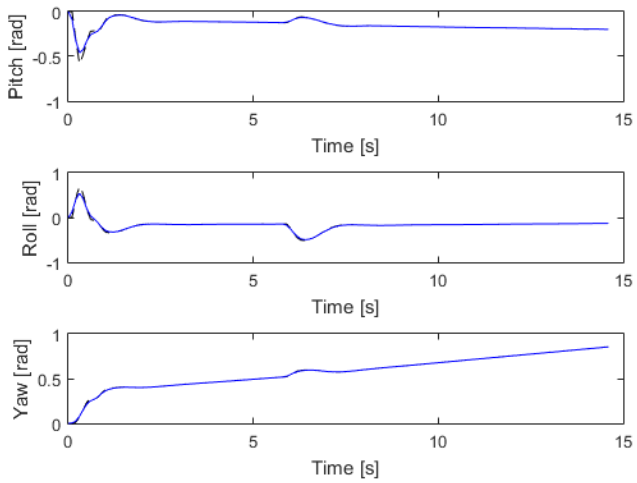

Figure 4.35: From top to bottom, evolution of ϕ, θ and ψ applying continuous wind disturbance and using parameters (4.14) with $N = 9$. Black dotted line: discrete computed values by the optimizer. Blue line: continuous real values

For higher values of N, the tendency of obtaining a higher admissible C_f is maintained, as well as the decrease in performance at those values of C_f. For instance, with $N = 10$ the maximum friction coefficient obtained is $C_f = 2500$. Performance results are shown in Table 4.13.

for error ε_i:	max absolute value (max $\lvert\varepsilon\rvert$)	mean value	standard deviation
x [m]	1.0249	0.8052	0.3057
y [m]	0.6107	-0.1288	0.1322
z [m]	0.2030	0.0762	0.0814

Table 4.13: Data measurements of the error signals for the simulation of the complete path tracking applying continuous wind disturbance and using parameters (4.14) with $N = 10$

In general, the controller seems able to provide robustness for relatively high values of C_f. If we consider the first simulation, in which $C_f = 520$, that friction coefficient value would probably correspond to a fluid with a low Reynolds number, meaning a high viscosity and low velocity [6]. So, in the situation of moving throw the air, we may assume that the friction coefficient is going to be lower, and therefore we can select the configuration with parameters (4.14) and $C_f = 520$ for the next simulations.
Now, using the mentioned configuration, it will be applied to a worst-case scenario in which, for a complete path tracking, it is going to be included the following wind disturbances:

$$\dot{x}_w = 20.8 \tag{4.15}$$
$$\dot{y}_w = 20.8sin(2\pi 0.1t)$$

The sinusoidal wind disturbance profile can be seen in Figure 4.36.

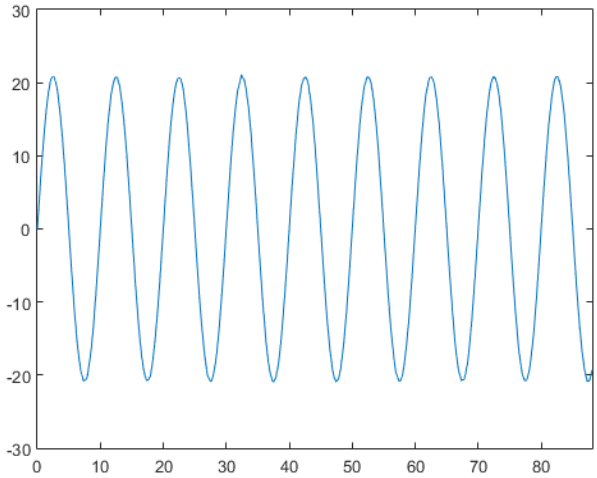

Figure 4.36: Sinusoidal wind profile of amplitude 20.8 and frequency 0.1 Hz

The first simulations are unable to finish, due to the optimization being infeasible at an early stage. Therefore, the velocity reference v is reduced to $v = 1.9$ m/s, which is the maximum linear velocity for which the simulation is feasible. With this value, the simulation is able to finish with $r_T = 4.8189$. The results obtained are shown in Table 4.14 and Figures 4.37 and 4.38. The simulation is completed with $r_T = 5.1154$.

for error ε_i:	max absolute value (max $\|\varepsilon\|$)	mean value	standard deviation
x [m]	0.4591	0.2478	0.0940
y [m]	0.4687	0.0181	0.1763
z [m]	0.2266	-0.0047	0.0467

Table 4.14: Data measurements of the error signals for the simulation of the complete path tracking using parameters (4.14) applying continuous and sinusoidal wind disturbances.

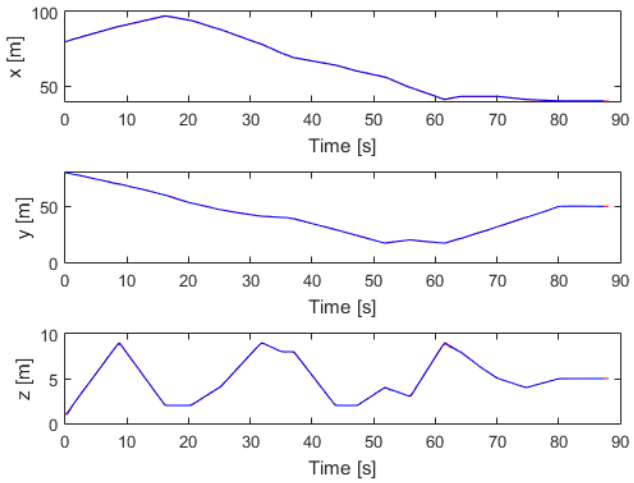

Figure 4.37: From and top to bottom, evolution of x, y and z applying continuous and sinusoidal wind disturbances and using parameters (4.14) with $N = 9$. Red line: reference values. Black dotted line: discrete computed values by the optimizer. Blue line: continuous real values

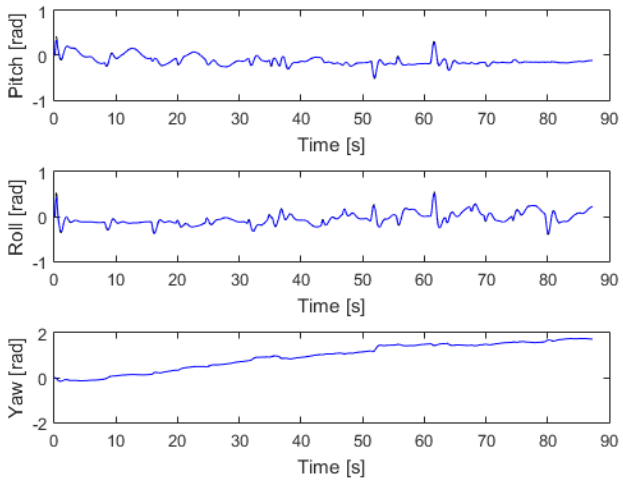

Figure 4.38: From top to bottom, evolution of ϕ, θ and ψ applying continuous and sinusoidal wind disturbances and using parameters (4.14) with $N = 9$. Black dotted line: discrete computed values by the optimizer. Blue line: continuous real values

The results of this simulation show that, despite the added disturbances, the NMPC controller is able to provide relatively high robustness in front of the considered wind distur-

bance values. Performance remains considerably good, as the steady state error in x due to the constant wind disturbance and the oscillations in y due to the sinusoidal wind disturbance are notably low.

4.5 Tuning for real-time execution

So far, all simulations have been completed with a time ratio $r_T > 1$, meaning that this controller cannot be applied to a real quadrotor, as it would require more time to compute the feedback law for a given time instant than the sampling time between that time instant and the next one. Given that real-time execution is a critical feature of a controller in order to have a real application, we will try to obtain real-time execution by increasing the sampling time T_s.

Considering simulation using parameters (4.14) and applying continuous and sinusoidal wind disturbances on the x-axis and y-axis, respectively, then the necessary T_s, theoretically, can be computed, knowing that the value of r_T in the previous simulation is $r_T = 4.7$, as:

$$T_s = 0.1 r_T = 0.47 \tag{4.16}$$

We may consider a rounded value such that $T_s = 0.5$. Note that increasing the sampling time would lead, in a real application to the NMPC losing ''reaction time" and might be under the danger of colliding with an obstacle not considered in the initial mapping. Additionally, the use of a high sampling time can also affect performance, and even stability, by making the real states to deviate more from the computed states by the controller (which, in addition, would probably increase computational time).

So, increasing T_s does not imply that the r_T will decrease. The approach of reducing v could be taken in order to minimize the effect of the friction, although the wind disturbances would affect in the same magnitude.

In fact, several simulations in which T_s is increased show an increasing computational time as well. Even considering no friction forces, the effect of the Hub forces and the propeller gyroscopic effect are enough to cause an increase in the computational time due to the unmodelled dynamics.

Therefore, other options should be explored in order to reduce computational time and obtain real-time control.

Chapter 5

Economic cost

The economic cost of the project is defined by considering the following:

- Cost due to earnings.
- Cost of the assets.
- Cost of expenses.

5.1 Cost due to earnings

This portion of the cost is computed based on a considered annual salary of 30000 €. The company signing on this project would have to pay a 30% of said salary to Spain's Social Security, so the total annual cost would be 39000 €. With this value, and knowing that year 2016 has 1769 working hours, the cost per hour is 22.05 €/hour. Therefore, the actual cost due to earnings comes in base to the number of hours required to finish the project, which are:

- Formation: 176 hours.
- Development of the project: 512 hours.
- Document writing: 184 hours.

The breakdown of this cost can be seen in Table 5.1.

	Number of hours	€/hour	Total Cost [€]
Formation	176	22.05	3880.8
Development	512	22.05	11289.6
Document	180	22.05	3969
TOTAL			19139.4

Table 5.1: Breakdown of costs due to earnings

5.2 Cost of assets

The assets employed for the development of this project are a PC and Matlab software. The PC has a value of 800 €, while the Matlab software requires a licensing for its use. The individual license has a value of 2000 €. No additional toolbox licenses were necessary.
In order to include the cost relative to these assets, an amortization period has to be defined. Since the PC is not exclusively used for the project, it can be assigned an amortization period such as its operating life. In this case, the PC is expected to last for 6 years. On the other hand, establishing an amortization period for a license, if it is not time-limited, can be less obvious. Generally, it depends on the expected use of the software. So, we may consider an amortization period equal to that of the PC.
The exact cost due to the amortization of the PC is computed considering the percentage of the year working hours that it has worked on the project.
The breakdown of these costs can be seen in Table 5.2

	Price [€]	cost/year	% of year used	Total Cost [€]
PC	800	133.33	49.07	65.43
Matlab license	2000		49.07	163.57
TOTAL				229

Table 5.2: Breakdown of the costs of assets

5.3 Cost of expenses

The cost of expenses basically consists on the energy consumption due to the computer working on the project. Knowing the percentage of the year that it has been working, the mean power consumption of the PC, as well as the price of electricity, the cost associated can be computed. Additionally, some travel expenses, as well as other miscellaneous costs, such as paper, ink, have been approximated. The breakdown can be seen in Table 5.3

	Power [W]	Work hours	Price energy [€/KWh]	Total Cost [€]
Energy	200	868.05	0.141033	24.48
Travel				20
Miscellaneous				5
TOTAL				49.48

Table 5.3: Breakdown of the costs of expenses

5.4 Total cost of the project

Adding up the three computed costs, the obtianed total cost of the project is shown in Table 5.4

	Cost [€]
Earnings	19139.4
Assets	229
Expenses	49.48
TOTAL	19417.88

Table 5.4: Total cost of the project

Chapter 6

Environmental, social and economic impact

Nowadays, there is an extensive and constantly increasing use of quadrotors, or, more generally, UAVs. These vehicles provide a wide set of possibilities in terms of applications, with the corresponding consequences in terms of environmental, social and economic impact. Therefore, being a proper path control the most basic feature of any application, then the impact these applications may have are, by extension, affected by the control of the UAV.

These vehicles, themselves, have low environmental impact in terms of contamination (theoretically zero, since it is an electric vehicle, but the energy sources would have to be taken into account). A proper control of the quadrotors may help reduce its power consumption. However, the main benefit it could provide comes at the hand of its possible applications. For instance, in the Introduction it was cited a research case consisting in the use of quadrotors for fire prevention. This example helps to understand the potential beneficial impact that UAVs can have for the environment.

With respect to the social impact, UAVs, in particular quadrotors, still have the main share of civilian use related to entertainment. Of course, the use of a quadrotor with a controller able to guarantee stability, as well as to provide robustness and good performance leaves a better experience for the user. Additionally, and on a more serious note, there are other uses of UAVs that can have a potentially big social impact, such as the case, stated in the Introduction, of the German consortium that successfully implemented a service to provide medicines to remote and low-accessible areas with the use of quadrotors.

Finally, regarding the economic impact, this is probably the aspect that provides a higher

development rate for UAVs (along with military applications). The potential benefits that the use of UAVs can have for a company is mainly due to the reduction in costs that, for a similar application using a manned vehicle, would be considerably higher. Particularly, distribution companies have the opportunity of offering a service to their clients that can be faster than the traditional ground transportation, while at the same time cutting in personnel cost.

Chapter 7

Conclusions and future work

In this project, it has been shown that the use of NMPC to control a quadrotor is able to guarantee stability and feasibility and provide a considerably good performance. This have been the case for simple reference inputs as well as for complete path tracking, and considering an ideal model for the quadrotor, as well as considering a real model with added effects with respect to the control model. Disturbances and unmodelled dynamics have shown that the controller is able to provide some robustness, effectively keeping at low values the effect of said disturbances and unmodelled dynamics under the considered controller parameters and reference velocity.

Summarizing, in terms of stability and performance, the simulation results seen so far, by using NMPC have been more than satisfying.

However, the major drawback of this approach remains after the conclusion of this project, which is the inability to obtain real-time control. Even after several attempts to minimize the time ratio r_T, the best obtained value, considering the unmodelled dynamics and wind disturbances, has been of 4.72. This is relatively far from a reasonable value, which should be low enough (lower than one) such that it would guarantee real-time control even after considering additional time delays due to communication, taking data from the sensors and processing it, etc.

Therefore, any future work should focus on obtaining real-time control. This could be approached by considering the use of a commercial and more powerful nonlinear optimization solver, such as Knitro from Artelys and/or the use of a PC with a more powerful microprocessor.

After that (and if) real-time control is obtained for the simulation considering the extended

system model with disturbances, other improvements could be made on the existing project, such as:

- Consider all sensors necessary to measure the states.

- If, for some states, there are not sensors available, consider the design of an observer.

- Perform localization using odometry and considering all the sensors tolerances, with the possibility of using landmarks to reset the localization errors.

- Perform online mapping using the corresponding sensors (such as IR or Ultrasounds).

- Finally, if, after considering the mentioned enhancements of the project, stability, feasibility, performance and real-time control are guaranteed, implement the controller in a real device.

Bibliography

[1] Ashfaq Ahmad Mian, W. D. Modeling and backstepping-based nonlinear control strategy for a 6 dof quadrotor helicopter. *Chinese Journal of Aeronautics 21*, 3 (2008), 261–268.

[2] Bouabdallah, S. *Design and control of quadrotors with application to autonomous flying*. PhD thesis, École Polytechnique Féderale de Laussane, 2007.

[3] Bouabdallah, S., and Siegwart, R. Backstepping and sliding-mode techniques applied to an indoor micro quadrotor. *International Conference on Robotics and Automation* (2005), 2247–2252.

[4] David J. Esteves, Alexandra Moutinho, J. R. A. Stabilization and altitude control of an indoor low-cost quadrotor: design and experimental results. *IEEE International Conference on Autonomous Robot Systems and Competitions* (2015), 150–155.

[5] Gary Fay, Peter Kunz, S. C. T. F. C. P. *The Mesicopter: A Miniature Rotorcraft Concept*. PhD thesis, Stanford University, 2001.

[6] Hoerner, S. F. *Fluid-Dynamic Drag: theoretical, experimental and statistical information*. Sighard F. Hoerner, 1965.

[7] Hossein Bolandi, Mohammad Rezaei, R. M. H. N. S. M. S. Attitude control of a quadrotor with optimized pid controller r. *Intelligent Control and Automation*, 4 (2013), 335–342.

[8] J.R. Martinez-de Dios, B.C. Arrue, A. O. L. M. F. G.-R. Computer vision techniques for forest fire perception. *Image and Vision Computing*, 26 (2015), 550—562.

[9] K. Alexis, G. Nikolakopoulos, A. T. Model predictive control scheme for the autonomous flight of an unmanned quadrotor. *IEEE International Symposium on Industrial Electronics* (2011), 2243–2248.

[10] K. Alexis, G. Nikolakopoulos, A. T. Model predictive quadrotor control: attitude, altitude and position experimental studies. *IET Control Theory and Applications 8* (2012), 1812––1827.

[11] Lars Grüne, J. P. *Nonlinear Model Predictive Control: Theory and Algorithms*. Springer, 2011.

[12] S. Bouabdallah, A. Noth, R. S. Pid vs lq control techniques applied to an indoor micro quadrotor. *1EEEIRS.J Internationel Conference On Intelligent Robots and Systems* (2004), 2451–2456.

[13] Watkinson, J. *The Art of the Helicopter*. Butterworth-Heinemann, 2003.